# Digital Footprinting, Analysis and Archival of Digital Media Content using Vision Transformers and Related Techniques

Sanket Anand
Student ID: 1140213

Thesis Report for
Master of Science in Computer Science

Liverpool John Moores University & upGrad
March 2024

# ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr. Kashav Ajmera for all his help and advice with this thesis. I would also like to thank my parents; without whose support this would have not been possible. I also appreciate all the support I received from the other students in my batch. Lastly, I would like to thank the LJMU and Upgrad for the studentship that allowed me to conduct this thesis. I would also like to say thanks to everybody involved behind the scenes at Upgrad or the University side for bringing this program, be it the student coordinators, the LJMU faculty for regular connects, or the markers for their transparent feedback and pointers on this study at all the stages of the program that has helped immensely in grooming this work in best possible manner as it stands today.

# ABSTRACT

Deep learning and point cloud processing have seen remarkable advancements in recent years with the advent of diffusion models and generative AI. The notion of a software system or what a typical software system should consist of has evolved with the advent of AI. In this work, we intend to propose a general purpose framework for building semi-autonomous software systems that are functional-reactive in nature and takes the advantage of recent advancements in deep learning and point cloud processing to ship downstream industrial applications that are relatively easier to build and works more like a living system that is capable of self-adjusting itself according to the data it is working on or the ever-evolving domain context. We build in this direction by first formalizing the general concept of a neural network into an abstract structure guided by point cloud tokens. We then use this construction to explore a machine learning model as a part of a semi-autonomous software system that can be verified using formal methods and can also be studied and analyzed in an academic setting using algebraic, topological and analytical methods. Finally, we then conclude the thesis work by demonstrating how this proposed abstract structure fits in the form of a data structure that is well-suited for all aspects of a software system such as data archival, foot-printing, editing and analysis using the capabilities of machine learning.

# TABLE OF CONTENTS

**ACKNOWLEDGEMENTS** ..... **2**
**ABSTRACT** ..... **3**
**LIST OF TABLES** ..... **7**
**LIST OF FIGURES** ..... **8**
**LIST OF ABBREVIATIONS** ..... **10**
**CHAPTER 1: INTRODUCTION** ..... **11**
1.1 Background of the Study ..... 11
1.2 Research Questions ..... 12
1.3 Aim & Objectives ..... 13
1.4 Significance of the Study ..... 13
1.6 Structure of the Study ..... 16
**CHAPTER 2: LITERATURE REVIEW** ..... **17**
2.1 Introduction ..... 17
2.2 Mathematical Fundamentals ..... 17
2.2.1 Introduction ..... 17
2.2.2 Abstract Algebra ..... 17
2.2.3 Real Analysis and Measure Theory ..... 20
2.3 Deep Learning ..... 21
2.3.1 Introduction ..... 21
2.3.2 Deep Feedforward Networks ..... 21
2.3.3 Convolutional Networks ..... 22
2.3.4 Recurrent and Recursive Networks ..... 22
2.3.5 Siamese Networks ..... 23
2.3.6 Autoencoders ..... 23
2.3.7 Deep Generative Models ..... 24
2.3.7.1 Boltzmann Machines ..... 24
2.3.7.2 Restricted Boltzmann Machines ..... 24
2.3.7.3 Variational Autoencoder ..... 25
2.3.7.4 Generative Adversarial Networks ..... 25
2.3.7.5 Diffusion Transformers ..... 26
2.3.7.6 Vision Transformers ..... 26
2.3.7.7 Neural Radiance Fields ..... 27
2.3.8 Kolmogorov-Arnold Network ..... 27
2.4 Point Cloud Processing ..... 29
2.4.1 Introduction ..... 29
2.4.2 Deep Learning Based Point Cloud Analysis ..... 29

2.4.2.1 Introduction..... 29
2.4.2.2 Classification and Segmentation.....29
2.5 Theory of Computation Fundamentals..... 31
2.5.1 Introduction..... 31
2.5.2 Finite State Machines and Regular Languages.....31
2.5.3 Context Free Languages and Pushdown Automata..... 32
2.5.4 Turing Machines and Undecidability.....33
2.6 Summary..... 35
**CHAPTER 3: RESEARCH METHODOLOGY..... 36**
3.1 Introduction.....36
3.2 Algorithms & Techniques.....36
3.2.1 Dynamic Graph CNN.....36
3.2.2 Vector Neuron based DGCNN (VN-DGCNN).....37
3.2.3 Tree of Clarifications (ToC) framework..... 38
3.2.4 Church Turing Thesis.....38
3.3 Methodology..... 39
3.3.1 End-to-End Pipeline.....39
3.3.2 Data Description..... 40
3.3.3 Data Preparation and Intended Usage..... 41
3.3.3.1 ModelNet40.....41
3.3.3.2 ASQA.....41
3.3.4 Implementation..... 42
3.3.4.1 Transform Data into Point Cloud Representation..... 43
3.3.4.2 Encode Point Cloud Data into Archival Form.....43
3.3.4.3 Answering Query upon Encoded Point Cloud Data.....43
3.3.4.4 Targeted Editing via Encoded Point Cloud Data.....44
3.3.5 Design Considerations.....44
3.3.6 Evaluation..... 46
3.4 Tools..... 47
3.4.1 Software.....47
3.4.2 Hardware..... 47
3.5 Summary..... 47
**CHAPTER 4: IMPLEMENTATION & SIMULATION.....48**
4.1 Introduction.....48
4.2 Data Preparation.....48
4.3 Tokenizer.....49
4.4 Data Transformer..... 49
4.5 Formal Language Specification..... 51
4.6 Compute Engine.....52

4.7 Sample Implementation..... 52
4.7.1 Demo URL..... 53
4.7.2 Main Screen..... 53
4.7.3 Creating a New Rule..... 59
4.7.4 Reporting and Alerts..... 61
4.7.5 Final Word..... 63
4.8 Summary..... 63
**CHAPTER 5: RESULTS & EVALUATION..... 64**
5.1 Introduction..... 64
5.2 Quantitative Evaluation..... 64
5.3 Qualitative Evaluation..... 67
5.3.1 Question Answer Reasoning..... 68
5.3.2 SO(3) equivalence..... 69
5.3.3 Data Structure Representation..... 70
5.3.4 Turing Completeness of the Language Specification..... 71
5.4 Summary..... 72
**CHAPTER 6: CONCLUSION & RECOMMENDATIONS..... 73**
6.1 Discussion and Conclusion..... 73
6.1.1 Formal Representation of Point Cloud Tokens..... 73
6.1.2 Data Structure representation for captured data..... 73
6.1.3 Expressive Power of the Formal Language Specification..... 73
6.2 Contribution..... 73
6.3 Recommendations..... 75
6.3.1 Implementation Aspects..... 75
6.3.2 Methodological Aspects..... 75
**REFERENCES..... 76**
**APPENDIX A: PROJECT CODE..... 82**
**APPENDIX B: RESEARCH PROPOSAL..... 99**

# LIST OF TABLES

Table 5.1: Classification comparison on ModelNet40 dataset

Table 5.2: Evaluation comparison of long-form QA on ambiguous questions on ASQA dataset

Table 5.3 Classification accuracy on ModelNet40 dataset (VN-architecture)

# LIST OF FIGURES

Figure 1.1: CES 2025 keynote by NVIDIA CEO Jensen Huang

Figure 2.1: PointNet architecture

Figure 2.2: PointNet++ architecture

Figure 3.1: Dynamic graph CNN model architecture

Figure 3.2: Tree of Clarifications (ToC) framework

Figure 3.3: Chomsky hierarchy

Figure 3.4: End-to-end pipeline

Figure 4.1: Frames per second (illustration)

Figure 4.2: Point cloud and other variants (illustration)

Figure 4.3: Data transformer implementation

Figure 4.4: Formal language definition schema

Figure 4.5: File directory structure

Figure 4.6: Home page screen

Figure 4.7: Sample upload of HTML file

Figure 4.8: Email alert for DFA rule hit

Figure 4.9: Email alert for DFA rule hit (contd.)

Figure 4.10: Editing of metadata using “Replace Text”

Figure 4.11: Sample image upload to OCR API

Figure 4.12: JSON upload for OCR API response

Figure 4.13: JSON replace for OCR API response

Figure 4.14: Create new DFA rule screen

Figure 4.15: DFA rule created detail page

Figure 4.16: User detail page

Figure 4.17: User edit screen

Figure 4.18: Insights screen

Figure 4.19: Report detail screen

# LIST OF ABBREVIATIONS

API: Application programming interface

C: Set of all complex numbers

CNN: Convolutional neural network

DFA: Deterministic finite automaton

DGCNN: Dynamic graph CNN

DGM: Deep Generative Model

DiT: Diffusion Transformer

GAN: Generative adversarial network

KAN: Kolmogorov-Arnold network

MLP: Multilayer perceptron

N: Set of all natural numbers

NeRF: Neural radiance field

OCR: Optical character recognition

P: Set of all irrational numbers

Q: Set of all rational numbers

R: Set of all real numbers

RBM: Restricted boltzmann machine

RNN: Recurrent neural network

SaaS: Software as a Service

SNN: Siamese neural network

VAE: Variational autoencoder

ViT: Vision transformer

VN-architecture: Vector neuron architecture

VN-DGCNN: Vector neuron based DGCNN

Z: Set of all integers

Z/nZ: Set of integers modulo n

# CHAPTER 1: INTRODUCTION

## 1.1 Background of the Study

According to a 2023 RAND report (Ryseff et al., 2023), more than 80% of AI projects fail due to various reasons such as miscommunication, vendor dependence, poor data quality, incorrect technical metrics, etc being some of the major reasons. This estimate is further validated by Gartner notes in (Stamford, 2018) that claims upto 85% of ML projects ultimately fail to deliver on their intended promises to the business. Another astonishing 2023 survey by Rexer Analytics (Rexer et al., 2023) found that only 32% of machine learning models successfully transition from pilot into production.

These problems can be attributed to the fact that to the best of our knowledge, there is no framework in existing literature that can be leveraged to not only bring all the stakeholders be it the technology team, business leadership, or the users to a common base, but can also be used to simulate and thoroughly verify the ML system being built during the early stages of the development itself. In this work, we attempt to cover this gap by proposing a robust framework that solves this problem by providing a concrete base on which we can potentially study and simulate the Machine Learning system as well as thoroughly conduct a formal verification of it before being pushed for a release. It can also be used for academic research for developing new Machine Learning software systems.

In recent years, deep learning and generative AI has seen major breakthroughs, some of them being diffusion models (Chang et al., 2023) and transformers (Vaswani et al., 2017). Today, it finds applications in diverse areas such as natural language processing, computer vision, multimodal AI, etc. just to name a few.

Another area that has seen massive strides in recent times is point cloud processing and its subsequent application into deep learning applications. This is one of the key concepts we will be leveraging later on in our framework. A very good coverage on this subject is found in (Liu et al. 2021).

Generative models, by nature are resource hungry and a lot of recent research has been focused on enabling them for edge devices. There has also been a good amount of effort in development

of completely new deep learning architectures fundamentally inspired from the same Transformer based architecture.

There has been a considerable amount of effort in advancing RAG Systems that improvise a Trained Model with domain specific or real time ground truth, thereby preventing hallucinations (Kim et al., 2023). This is a critical component we intend to account for in our framework later on, as modern software systems need to be essentially considered as a living system that evolves with changing data dynamics.

In the domain of media analytics, there has been substantial progress around 3D reconstruction of Images and media content using generative AI Techniques. Good amount of research has been done around related downstream applications such as depth estimation (Yang et al., 2024), temporal spatial reasoning (Cheng et al., 2024), pose estimation (Dwivedi et al., 2024), virtual try on (Zhu et al., 2024), etc just to name a few. We consider this as noteworthy mention because the framework we will be proposing aims to be generic in nature capable of working on any type of unstructured data- audio, video, image, text, etc as one would expect in industrial use cases such as call center logs, emails, video recordings, scanned documents, etc.

There has also been a considerable amount of work in related areas such as explainable AI, graph based RAG systems (Edge et al., 2024), Spreadsheet LLM (Tian et al., 2024), etc. Again, with an aim to be the one size fits all framework, we believe these are noteworthy additions in dealing with data in the form of spreadsheets, graphs, etc with RAG support.

## 1.2 Research Questions

This thesis tries to answer the following questions:

1) Can we accurately model the stream of tokens from our neural network into an abstract structure?

2) What is the most accurate abstract structure that can correctly capture the token value as well as its underlying relationship with other tokens.

3) Does this abstract structure respect results from Church Turing thesis in the sense it is equivalent to a Turing recognizable language?

4) Can we subsequently implement this abstract structure in the form of a data structure for media analytics applications?

## 1.3 Aim & Objectives

This work aims to propose a complete framework for building AI enabled software systems that supports digital footprinting, analysis, archival and editing of digital media content such as audio, video, text, etc.

Our intended objectives out of this work are:

1. To formalise the notion of a neural network into an abstract structure while correctly capturing all the underlying properties.
2. To correctly map the abstract structure in the form of a data structure for media content archival and editing.
3. To formally evaluate if the aforementioned abstract structure is Turing recognizable.
4. To conduct a literature review of developments happening around deep learning in recent years.

## 1.4 Significance of the Study

In the recently concluded keynote at CES 2025 Expo (Caulfield, B., 2025), NVIDIA CEO Jensen Huang asserted the fact that industry is now moving focus into physical AI where AI can now plan and act on its own in real time. While there has been a real evolution from Perception AI where the focus of the industry was to make the computer understand the world such as “Speech to Text”, “Video to Text”, etc, we gradually progressed into Generative AI that can also generate similar content in various modalities such as audio, video, image, text, etc. on its own.

Post this, the industry was quick to shift focus on Agentic AI which is basically making computers learn and act on its own such as process automation, software development assistants, etc. Now, focus has rather quickly been shifting into physical AI applications such as Self Driving Cars, Robotics, etc.

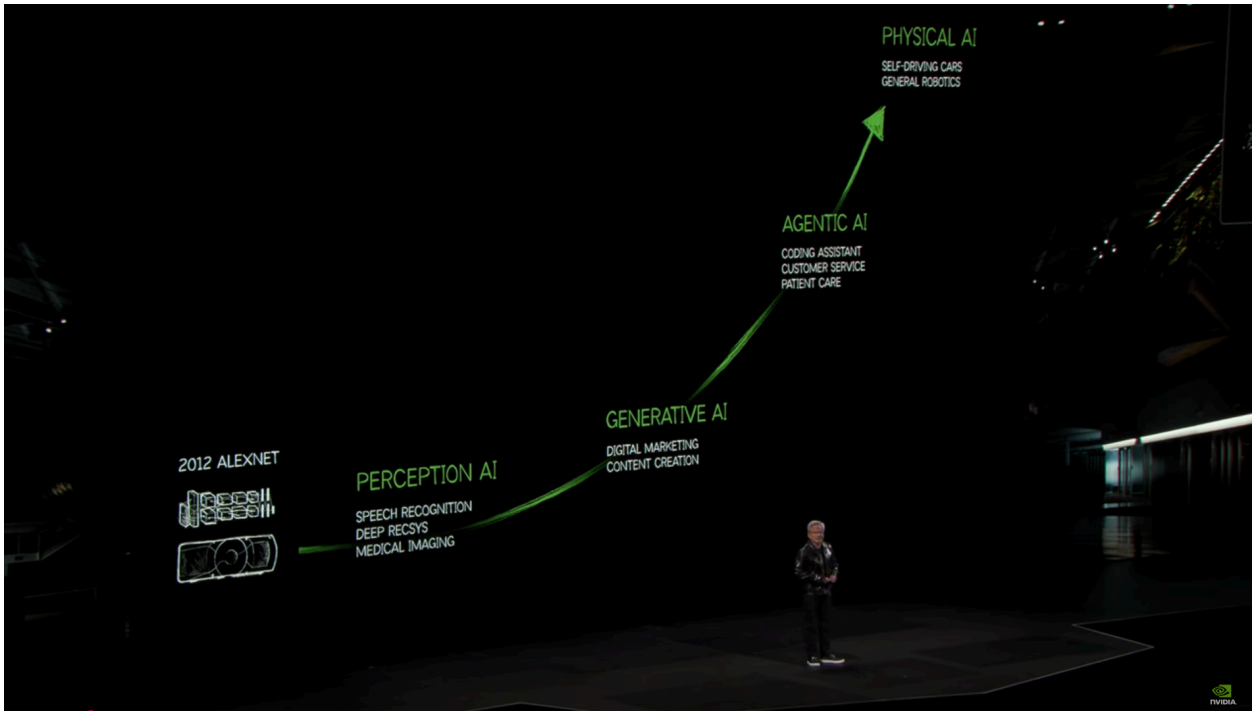


Figure 1.1: CES 2025 keynote by NVIDIA CEO Jensen Huang (Caulfield, B., 2025)

It is apparent that Deep Learning and AI in particular is a field under active research and has completely changed the way we compute or work in a computing sense with a machine. All of this started with the landmark release of Transformer in the 2017 Google Paper (Vaswani et al., 2017). While the current methods rely on considering a neural network as a system of Bayesian probabilities, there has been little to no literature work that has happened around exploring the same from an analytical perspective in an equivariant vector form.

This holds significance because despite the progress being made, there is also a very large number of failed projects or projects that are unable to meet the business requirements being happening as discussed in (Ryseff et al., 2023; Stamford, 2018; Rexer et al., 2023). This work tries to fill in this gap by introducing methods and techniques exploring the field of Machine Learning and AI from a pure mathematics lens. This study aims to be a standard framework for analyzing a generic AI enabled software system with CRUD(Create, Read, Update, Delete).

In terms of applications, we then show how the proposed structure can also be implemented as an efficient data structure for archival and editing of media content in AI use cases.

We will propose a framework built on point cloud architecture that can be used for end to end operations on all forms of data such as classification, rapid prototyping, spatial and temporal reasoning as well as data driven intelligent systems. To the best of our knowledge, there is no such framework in existing literature that can be used to study, evaluate, and verify an AI implementation project, and this is precisely the gap we look forward to covering in this project. All the existing software systems consist of a database where typically the user stores data in the form of a record, document or a row.

This framework allows the user to directly capture and work upon required insights from unstructured data points such as call center logs, emails, surveillance footage, etc in a typical use case and will also facilitate in academic study, formal verification and development of such software systems. As an example, consider a full stack software application installed at a service center providing on-call, chat as well as email support. Here, a swarm lead for instance, would like to know how many support cases have been received for a newly launched product, as well as segment type of queries, be it bugs around existing functionality, feature request, etc. The framework by nature directly pulls insights from unstructured data or end user data points such as chat history, email logs, call recordings, etc, pushes them into a schema-less data lake for future retrieval, and also answers any such existing user query.

The core strength of this framework is that being streaming in nature, there is no need to configure a schema, which massively reduces time to complete the implementation.

We will then proceed and demonstrate how this framework can be applied in general to any modern day software system project and highlight the benefits that we will get from such an approach. In terms of project scoping, consider the amount of time or budget that goes into schema design and maintenance on average. We will demonstrate how this framework can be applied to achieve a schema-less architecture that is easier to maintain for the end user and is instead driven by a rules engine.

To conclude, we start by investigating methods and techniques in existing literature to obtain a generic mathematical representation that models all types of media data- audio, video, image, text, and then study how it can be applied in the form of a design pattern to any software project.

## 1.5 Scope of the Study

The scope of this thesis work has been defined as follows:

- I have completed this thesis work in 3 phases or sprints of 1 month each post research proposal submission in an iterative manner with 3-4 mini-iterations in each sprint.
- There have been a total of 10+ such mini-iterations to complete this work.
- The quality of work has been subsequently improved with each sprint with stronger arguments and refinements.
- The sample implementation has been done on a free to use Salesforce dev org.
- Additional experiments and proof-of-concepts have been conducted on a local system with configuration: i7 13th Gen, 16 GB ram, 1 TB SSD, and RTX 3060 6GB GPU.
- The emphasis has been to build upon existing literature and express our results through a combination of both theoretical and practical techniques.

## 1.6 Structure of the Study

This study has been structured in the following manner:

- Chapter 1 – Introduction: This chapter provides introduction and background for this research work and sets the initial context including the motivation behind this study.
- Chapter 2 – Literature Review: This chapter covers the literature concepts needed to understand and apply the framework or the techniques discussed in this work.
- Chapter 3 – Methodology: This chapter discusses the research methodology and elaborates on the reasoning behind the techniques used in the proposed framework.
- Chapter 4 – Implementation & Simulation: This chapter discusses the low level implementation details of the proposed framework including a sample implementation for the proposed framework.
- Chapter 5 – Results & Evaluation: This chapter covers a qualitative and quantitative evaluation of the proposed framework and how it has helped in achieving the stated research goals.
- Chapter 6 – Conclusion & Recommendations: This chapter concludes with a brief discussion on what we have been able to achieve and future directions of research.

# CHAPTER 2: LITERATURE REVIEW

## 2.1 Introduction

This chapter discusses the existing literature that will serve as a base for the research work that has been carried out in this thesis report. Although some of the results over here are very fundamental in nature, the idea is to have all the relevant literature content for the work presented in this thesis in one place and self-contained fashion.

## 2.2 Mathematical Fundamentals

### 2.2.1 Introduction

Much of the deep learning and theoretical computer science concepts discussed in this work makes use of results from both pure as well as applied mathematics and thus, it is important to cover those before we dig deeper into the research methodology. We briefly cover concepts from Algebra, Analysis and Measure Theory in general that are absolutely critical for our proposed research work. For Abstract Algebra, the readers are suggested to refer (Dummit et al., 2004) for a more detailed coverage of the subject. Similarly, for Linear Algebra, the readers are suggested to refer (Cooperstein et al., 2015). For real analysis and measure theory (Royden et al., 2023) is a good reference, whereas for probability and statistics, readers are again requested to refer (Degroot et al., 2014) for further discussion on the subject.

### 2.2.2 Abstract Algebra

The basic building block wherein we study algebraic structures, which are basically sets with specific operations acting upon their elements. It is a fundamental fact that all the systems studied are sets, to which the theorem of set theory and the whole rest of results from mathematics apply, be it Pure Mathematics (Real Analysis, Topology, Measure Theory, Functional Analysis, Algebra, etc) or Applied Mathematics (Probability and Statistics, Complex Analysis, Numerical Analysis, Calculus, etc). Some of the fundamental algebraic structures in Abstract Algebra include the following-

- **Group**- A group (Dummit et al., 2004, p. 16) is an ordered pair (G, *) where G is a set and * is a binary operation on G that satisfies the associative property, closed under

inverse for every element in G, and consists of a Identity Element. If the group is also satisfies the commutative property under *, it is said to be an abelian group.

For example, Z, Q, R and C are groups under + with e = 0 and $a^{-1} = 1a$, for all a.

Additionally, The subset H of G is a subgroup of G if H is nonempty and H is closed under product and inverse (i.e., $x, y \subset H$ implies $a^{-1} \in H$ for all values of a and $xy \in H$). If H is a subgroup of G we shall write $H \subseteq G$.

- **Ring**- A ring (Dummit et al., 2004, p. 223) R is a set together with two binary operations + and * (called addition and multiplication) satisfying the following axioms:

  1. (R, +) is an abelian group,

  2. * is associative: (a * b) * c = a * (b * c) for all $a, b, c \in R$,

  3. The distributive laws hold in R : for all $a, b, c \in R$,

     (a + b) * c = (a * c) + (b * c) and a * (b + c) = (a * b) + (a * c).

  Additionally, the ring R is commutative if multiplication is commutative.

  Also, the ring R is said to have an identity (or contain a '1') if there is an element

  with the property-

  1 * a = a * 1 = a  for all $1 \in R$

  For example, the ring of integers Z, under the usual operations of addition and multiplication is a commutative ring with identity (integer 1). Similarly, the rational numbers, Q, the real numbers, R, and the complex numbers, C, are commutative rings with identity.

- **Module**- Let R be a ring (not necessarily commutative nor with 1). A left R-module (Dummit et al., 2004, p. 337) or a left module over R is a set M together with-

  1. A binary operation + on M under which M is an abelian group, and

2. An action of R on M (that is, a map R * M → M) denoted by rm, for all $r \in R$ and $m \in M$ which satisfies-

i. (r + s)m = rm + sm, for all $r, s \in R, m \in M$,

ii. (rs)m = r(sm), for all $r, s \in R, m \in M$,

iii. r(m + n) = rm + rn, for all $r \in R, m, n \in M$.

3. Additionally, if the ring has a 1, we impose the additional axiom:

1m = m, for all $m \in M$.

Note that the descriptor 'left' in the above definition indicates that the ring elements appear on the left; "right" R-modules can be defined analogously. If the ring R is commutative and M is a a left R-module, we can make M into a right R-module by defining mr = rm for all $r \in R, m \in M$.

As an example, the quotient group Z/nZ is a Z-module.

- **Field**- A field (Dummit et al., 2004, p. 34) is a commutative ring with identity in which every non-zero element has an inverse. Equivalently, the set $F^x$ = F – {0} of non-zero elements of F is an abelian group under multiplication.

  For example, the set of rational numbers R and the set of complex numbers C are fields.

- **Vector Space**- (Dummit et al., 2004, p. 408) When the ring R is a field, we define the R-module M as a vector space over R. For M to be a vector space, the sole requirement is that R must be a field and M must be a R-module.

  As an example, the function that takes natural numbers to real numbers is a vector space. Addition and scalar multiplication are defined as adding or multiplying the functions together.

### 2.2.3 Real Analysis and Measure Theory

1. **Lebesgue Measure**- A collection of subsets of R is called σ-algebra (Royden et al., 2023, p.30) if it contains R and is closed with respect to compliment and countable union. By De Morgan's identities, such a collection is also closed with respect to countable intersections. Then, we define outer-measure m* as a set-function such that-

$$m^*(E) \leq m^*\left(\bigcup_{k=1}^{\infty} E_k\right) \leq \sum_{k=1}^{\infty} m^*(E_k)$$

   Here, a few properties are-

   a) The outer-measure of an interval is its length.

   b) Outer-measure is translation invariant.

   c) Outer-measure is countably additive over countable, disjoint union of sets.

   The collection of Lebesgue measurable sets is a σ-algebra that contains all open sets, and all sets of outer-measure zero. The Lebesgue measure 'm' is defined as a countably additive set function obtained by restricting m* to the collection of Lebesgue measurable sets.

2. **Measure Spaces**- A set-function $\mu : M \rightarrow \{0, \infty]$ on a σ-algebra M of subsets of X is called a measure provided that $\mu(\phi) = 0$ and it is countably additive, in the sense that for any countable, disjoint collection $\{E_k\}$ from k = 1 to ∞ of sets in M satisfies the following condition:

$$\mu\left(\bigcup_{k=1}^{\infty} E_k\right) = \sum_{k=1}^{\infty} \mu(E_k)$$

   The triple (X, M, μ) is called a measure space (Royden et al., 2023, p.159) and a set in M is said to be measurable.

3. **Metric Spaces**- Let X be a non-empty set. A function p: X * X → R is called a metric provided that for all x, y and z in X,

   a) $p(x, y) \geq 0$

b) $p(x, y) = 0$ if and only if x = y;

c) $p(x, y) = p(y, x)$

d) $p(x, y) \leq p(x, z) + p(z, y)$. (Also called as triangle inequality)

The set X together with the metric 'p' is called a metric space (Royden et al., 2023, p.263) and is denoted by (X, p).

## 2.3 Deep Learning

### 2.3.1 Introduction

The concepts briefly discussed in this section builds upon the mathematical concepts covered so far and formally defines the machine learning systems that are relevant to our research work later on. For a more detailed reference, readers are requested to refer (Goodfellow et al., 2016).

### 2.3.2 Deep Feedforward Networks

Deep Feedforward networks (Goodfellow et al., 2016, p.163), also called feedforward neural networks, or multilayer perceptrons (MLPs) are a type of artificial neural network where connections between the nodes do not form cycles. The network consists of an input layer, one or more hidden layers, and an output layer. Information flows in one direction- from input to output- hence the name 'feedforward'. The goal of a feedforward network is to approximate some function $f^*$. For example, for a classifier, $y = f^*(x)$ maps an input 'x' to a category 'y'. Some applications of these type of neural networks are:

- Pattern recognition.
- Classification tasks.
- Regression analysis.
- Image recognition.
- Time series prediction.

### 2.3.3 Convolutional Networks

Convolutional networks (Goodfellow et al., 2016, p.321), also known as convolutional neural networks, or CNNs, are a specialized kind of neural network for processing data that has a known grid-like topology.

Examples include time-series data, which can be thought of as a 1-D grid taking samples at regular time intervals, and image data, which can be thought of as a 2-D grid of pixels.

In simpler terms, Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers. Here, convolution is defined as a specialized kind of linear operation as:

$$s(t) = \int x(a)w(t - a)da.$$

This operation precisely is called convolution. The convolution operation is typically denoted with an asterisk:

$$s(t) = (x * w)(t).$$

Some applications of CNNs are:

- Computer Vision tasks such as image recognition, object tracking, etc.
- Video Analytics tasks such as action detection, video scene segmentation, etc.
- Natural Language Processing tasks such as sentiment analysis, language translation, etc.

### 2.3.4 Recurrent and Recursive Networks

Recurrent neural network (RNN) (Goodfellow et al., 2016, p.368) is a class of neural network designed for processing sequential data or a sequence of values $x^{(1)}, \dots, x^{(r)}$ and is particularly useful in scenarios where we need to capture time dependencies. Unlike feedforward neural networks, RNNs have connections that create loops within the network, allowing them to maintain a form of memory.

Some applications of RNNs are:

- Language Modeling.
- Time series prediction.

Now, recursive networks, on the other hand, are designed to handle hierarchical structures, making them particularly suited for tasks involving tree-like or nested data. These networks explicitly model relationships and dependencies in hierarchical arrangements, such as syntactic structures in languages or hierarchical representations in images.

Some applications of recursive networks are document structure analysis, image parsing, etc.

### 2.3.5 Siamese Networks

A Siamese neural network (SNN) is a class of neural network architectures that contain two or more identical sub-networks. “Identical” here means they have the same configuration with the same parameters and weights. Parameter updating is mirrored across both sub-networks and it’s used to find similarities between inputs by comparing feature vectors. For further reference, readers are requested to refer (Chicco, 2020).

Some applications are:

- Few shot classification tasks.
- Facial recognition tasks.

### 2.3.6 Autoencoders

An Autoencoder (Goodfellow et al., 2016, p.493) is a neural network that is trained to attempt to copy its input to its output. Internally, it has a hidden layer h that describes a code used to represent the input. The network may be viewed as consisting of two parts: an encoder function h = f(x) and a decoder that produces a reconstruction r = g(h). It maps an input ‘x’ to an output (called reconstruction) ‘r’ through an internal representation or code ‘h’. The autoencoder has two components: the encoder ‘f’ (mapping ‘x’ to ‘h’) and the decoder ‘g’ (mapping ‘h’ to ‘r’).

Applications for autoencoders include facial recognition, feature detection, noise elimination from images, etc.

### 2.3.7 Deep Generative Models

Deep generative models (DGMs) (Goodfellow et al., 2016, p.645) are a type of artificial intelligence (AI) and machine learning (ML) technique that uses deep neural networks to learn the distribution of complex data sets. They can generate large datasets that can predict the probability of the next item in a sequence.

Some applications of DGMs are Text to Image Synthesis, Text to Speech Synthesis, Temporal-Spatial Reasoning, etc.

We briefly cover the major developments that have taken place in the field of deep generative models.

#### 2.3.7.1 Boltzmann Machines

We define the Boltzmann Machine (Goodfellow et al., 2016, p.645) over a d-dimensional binary random vector $x \in \{0, 1\}^d$ . The Boltzmann machine is an energy-based model meaning we define the joint probability distribution using an energy function:

$$P(x) = \frac{exp(-E(x))}{Z}$$

where E(x) is the energy function, and Z is the partition function that ensures that $\Sigma_x$ P(x) = 1. The energy function of the Boltzmann machine is given by:

$$E(x) = -x^T U x - b^T x,$$

where U is the "weight" matrix of model parameters and 'b' is the vector of bias parameters.

#### 2.3.7.2 Restricted Boltzmann Machines

Let the observed layer consist of a set of $n_v$ binary random variables, which we refer to collectively with the vector v. We refer to the latent, or hidden, layer of $n_h$ binary random variables as h.

The Restricted Boltzmann Machine (RBM) (Goodfellow et al., 2016, p.647) is an energy-based model with the joint probability distribution specified by its energy function:

$$P(\mathrm{v} = v, \mathrm{h} = \mathrm{h}) = \frac{1}{Z} exp(- E(v, \mathrm{h})).$$

The energy function for an RBM is given by:

$$E(v, \mathrm{h}) = -\mathrm{b}^{\mathrm{T}} v - \mathrm{c}^{\mathrm{T}}\mathrm{h} - v^{T} W \mathrm{h},$$

and Z is the normalizing constant known as the partition function:

$$Z = \sum_{v} \sum_{\mathrm{h}} \exp \{- E(v, \mathrm{h})\}.$$

#### 2.3.7.3 Variational Autoencoder

The Variational Autoencoder (VAE) (Kingma et al., 2014) is a directed model that employs learned approximate inference and can be trained entirely using gradient-based methods. To generate a sample, the VAE first samples z from the code distribution $p_{model}(z)$. This sample is then passed through a differentiable generator network g(z). Next, x is sampled from the distribution $p_{model}(x; g(z)) = p_{model}(x \mid z)$. During training, the approximate inference network (also known as the encoder) $q(z \mid x)$ is used to obtain z, and $p_{model}(x \mid z)$ functions as the decoder network.

#### 2.3.7.4 Generative Adversarial Networks

Generative Adversarial Networks (GANs) (Goodfellow et al., 2016, p.690) operate on a game-theoretic framework where the generator network competes against an adversarial network. The generator network creates samples directly, represented as $x = g(z; \theta^{(g)})$. Its adversary, the discriminator network, tries to differentiate between samples from the training data and those produced by the generator. The discriminator outputs a probability, $d(x; \theta^{(d)})$, which indicates the likelihood that x is a real training sample rather than a synthetic one generated by the model.

Effectively, GANs work in principle by minimizing the loss function $J_G$ defined as:

$$J_G = \frac{-1}{m} \sum_{i=1}^{m} logD(G(z_i))$$

Where, $J_G$ measures how well the generator is fooling the discriminator,

Log $D(G(z_i))$ represents log probability of the discriminator being correct for generated samples,

The generator aims to minimize this loss, encouraging the production of samples that the discriminator classifies as real (log $D(G(z_i))$), close to 1.

Some applications of GANs include image generation, video generation, 3D object generation, etc just to name a few.

**2.3.7.5 Diffusion Transformers**

Diffusion Transformer (DiT) (Chang et al., 2023) is a class of diffusion models that are based on the transformer architecture. Developed by William Peebles at UC Berkeley and Saining Xie at New York University, DiT aims to improve the performance of diffusion models by replacing the commonly used U-Net backbone with a transformer.

DiT uses transformers in a latent diffusion process, where a simple prior (like Gaussian noise) is gradually transformed into the target image. This is done by reversing the diffusion process guided by a transformer network. An important aspect of DiT is the concept of diffusion time-steps. These time-steps represent the stages of the diffusion process, and the transformer network is conditioned on the time-step at each stage. This allows the network to generate different features at different stages of the diffusion process. DiT can also be conditioned on 'class labels', allowing it to generate images of specific classes.

**2.3.7.6 Vision Transformers**

Vision Transformer (ViT) (Dosovitskiy et al., 2021) uses transformers to directly generate the image in an auto-regressive manner, where each patch is generated one after the other,

conditioned on the previously generated patches. A key component of ViT is the use of adaptive layer norm layers (adaLN).

These layers adaptively scale and shift the features based on the statistics of the current batch, which helps in stabilizing the training and improving the model's performance. While both approaches have their strengths and weaknesses, they represent two promising directions for leveraging transformers in generative modeling of images. The choice between DiT and ViT would depend on the specific requirements of the task at hand.

For instance, if the task requires generating images of specific classes, DiT might be a better choice due to its ability to condition on class labels. On the other hand, if the task requires generating high-resolution images, ViT might be more suitable due to its use of adaLN layers, which can help in stabilizing the training of large models.

#### 2.3.7.7 Neural Radiance Fields

A neural radiance field (NeRF) (Mildenhall et al., 2020) is a type of neural network designed to reconstruct intricate 3D scenes from a limited set of 2D images. 3D images are essential in various fields such as simulations, gaming, media, and Internet of Things (IoT) applications to enhance digital interactions with greater realism and precision. NeRF learns the geometry, objects, and perspectives of a scene, enabling it to generate photorealistic 3D views from new angles and automatically produce synthetic data to cover any missing information.

The NeRF algorithm represents a scene as a radiance field parametrized by a deep neural network. The network predicts a volume density and view-dependent emitted radiance given the spatial location (x, y, z) and viewing direction in Euler angles (θ, Φ) of the camera. By sampling many points along camera rays, traditional volume rendering techniques can produce an image.

NeRF finds applications in areas such as computer graphics and animation, 3D reconstruction, medical imaging, virtual reality, etc.

### 2.3.8 Kolmogorov-Arnold Network

A Kolmogorov-Arnold Network (KAN) (Liu et al., 2024) is grounded in a theorem proposed by Andrey Kolmogorov and later expanded by Vladimir Arnold. This theorem asserts that any

multivariate continuous function can be expressed as a sum of continuous functions, each depending on just one variable. This result is a key principle in the theory of neural networks, particularly in understanding their ability to perform universal approximation.

In formal terms, for any continuous function f with n variables, there exist continuous functions $\phi$ and $\psi$ such that:

$$f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{2n+1} \phi_i \left( \sum_{j=1}^{n} \psi_{ij}(x_j) \right)$$

KANs aim to utilize this theorem to address the curse of dimensionality, which often complicates high-dimensional problems in machine learning. By placing learnable activation functions on the edges rather than the nodes, KANs can require fewer parameters to effectively approximate high-dimensional functions compared to traditional MLPs.

Some key characteristics of KANs include:

- While MLPs have fixed activation functions on nodes ("neurons"), KANs feature learnable activation functions on the edges ("weights").
- KANs eliminate linear weights entirely, replacing them with univariate functions parameterized as splines. This modification enhances both the accuracy and interpretability of KANs compared to MLPs.

In summary, KANs can achieve similar or even better accuracy than larger MLPs for tasks like data fitting and solving partial differential equations (PDEs). They exhibit faster neural scaling properties, both theoretically and empirically, and their structure is more intuitive, making them easier to interpret and interact with for human users.

## 2.4 Point Cloud Processing

### 2.4.1 Introduction

A 3D point cloud consists of a collection of points in three-dimensional space, representing the surface of an object. Each point is defined by three coordinates that specify its position relative to three orthogonal axes. Additionally, extra attributes like RGB color values and surface normals can be included depending on the sensor capturing the point cloud. This section briefly introduces point cloud technology and highlights advancements in both traditional and deep learning-based point cloud analysis. For further reference, readers are suggested to refer (Liu et al., 2021) for a detailed discussion on the subject of 3D point cloud analysis.

### 2.4.2 Deep Learning Based Point Cloud Analysis

#### 2.4.2.1 Introduction

In recent years, deep learning has significantly outperformed traditional methods in point cloud processing tasks. It offers substantial advantages, including distributed processing capabilities and notable improvements in performance and benchmarking, as outlined in the following discussion.

#### 2.4.2.2 Classification and Segmentation

- **PointNet**: PointNet (Qi et al., 2017b, Liu et al., 2021) was the first approach to apply deep learning directly to 3D point clouds.

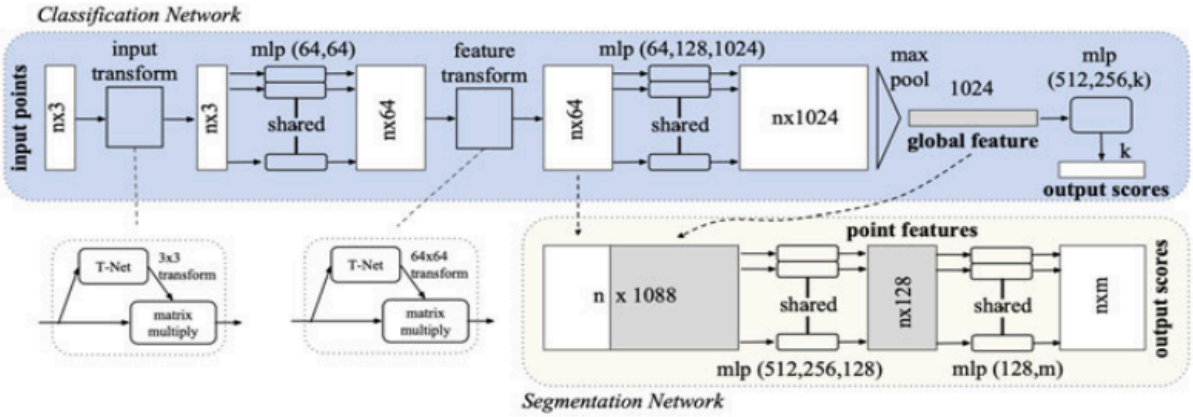


Figure 2.1 : PointNet architecture (Qi et al., 2017b)

Before its introduction, 3D point clouds were typically converted into regular formats like voxel grids or multiple 2D projections for processing.

- **PointNet++**: PointNet delivers outstanding results in point cloud processing tasks such as classification and segmentation. However, it lacks the ability to capture multi-scale local context information of points. To address these limitations, a subsequent approach called PointNet++ (Qi et al., 2017c, Liu et al., 2021) introduces a hierarchical feature learning framework. This framework utilizes multiple set abstraction levels, each comprising a sampling layer, grouping layer, and PointNet layer, to enable more effective learning of local and global point cloud features.

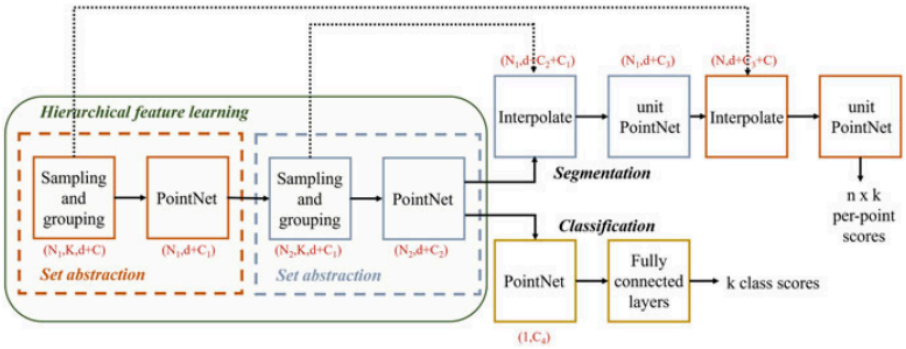


Figure 2.2 : PointNet++ architecture (Liu et al., 2021, p. 56)

- **Dynamic Graph CNN (DGCNN)**: DGCNN (Wang et al., 2019) is an innovative approach that employs EdgeConv operators to effectively capture local neighborhood information within point clouds.

  The output of EdgeConv is calculated by aggregating the edge features associated with edges from each connecting vertex. EdgeConv operates on graphs constructed from the point cloud at each layer of the network. By cascading multiple layers of EdgeConv, the network learns global shape features. Additionally, DGCNN dynamically updates the graph structure across layers, enabling semantic grouping of points.

  As we will discuss this in detail in the next chapter, DGCNN has been adopted in multiple point cloud processing related frameworks. We will also explore how it is relevant to our work and build our framework on it.

## 2.5 Theory of Computation Fundamentals

### 2.5.1 Introduction

Having covered the mathematical base along with deep learning concepts and point cloud processing, we are now going to briefly introduce the final piece that binds all of these into a rigorous framework that can be used for various applications such as development of deep learning based point cloud analysis techniques, simulation, as well as formal verification of deep learning systems. For a thorough discussion on automata theory, readers are requested to refer (Rich, 2008) which covers pretty much all of the theory of computation topics in a great depth that are used in this study.

### 2.5.2 Finite State Machines and Regular Languages

A finite state machine (Rich, 2008, p.54) (FSM) is a computational model that takes a string of inputs and produces one of two outcomes: Accept or Reject. An input string is processed by the FSM character by character, moving from left to right. With each new character, the FSM evaluates its current state and transitions to a new state. Some of the states may be designated as accepting states.

A deterministic finite automaton M is a 5-tuple, (Q, Σ, δ, q0, F), consisting of:

- A finite set of states Q
- A finite set of input symbols called the alphabet Σ
- A transition function $\delta : Q \times \Sigma \rightarrow Q$
- An initial or start state $q0 \in Q$
- A set of accept states $F \subseteq Q$

Here, the finite automaton M is called as non-deterministic if the transition function δ is defined as:

$Q \times \Sigma \rightarrow P(Q)$, where P(Q) is the power set over Q.

If the FSM runs out of input while in an accepting state, it accepts the string. If the FSM completes its processing and is not in an accepting state, it rejects the string. The FSM always halts after exactly |w| steps of processing.

Additionally, the set of languages accepted by some finite state machine is precisely referred to as a set of regular languages.

### 2.5.3 Context Free Languages and Pushdown Automata

A context-free grammar G (Rich, 2008, p.207) is defined by the 4-tuple G = (V, Σ, R, S), where

- V is a finite set; each element $v \in V$ is called a non-terminal character or a variable. Each variable represents a different type of phrase or clause in the sentence. Variables are also sometimes called syntactic categories. Each variable defines a sub-language of the language defined by G.
- Σ is a finite set of terminals, disjoint from V, which make up the actual content of the sentence. The set of terminals is the alphabet of the language defined by the grammar G.
- R is a finite relation in V x $(V \cup \Sigma)^*$, where the asterisk represents the Kleene star operation. The members of R are called the (rewrite) rules or productions of the grammar. (also commonly symbolized by a P)
- S is the start variable (or start symbol), used to represent the whole sentence (or program). It must be an element of V.

In formal language theory, a context-free language (CFL), also called a Chomsky type-2 language, is a language generated by a context-free grammar (CFG). We use standard formal language notation: $\Gamma^*$ denotes the set of finite-length strings over alphabet Γ and ε denotes the empty string.

A PDA (Rich, 2008, p.249) is formally defined as a 7-tuple M = $(Q, \Sigma, \Gamma, \delta, q_0, Z, F)$ where:

- Q is a finite set of states
- Σ is a finite set which is called the input alphabet
- Γ is a finite set which is called the stack alphabet

- δ is a finite subset of Q x (Σ ∪ {ε}) x Γ x Q x $\Gamma^*$ , the transition relation
- $q_0 \in Q$ is the start state
- $Z \in \Gamma$ is the initial stack symbol
- $F \subseteq Q$ is the set of accepting states

An element (p, a, A, q, α) ∈ δ is a transition of M. It has the intended meaning that M, in state p ∈ Q, on the input α ∈ Σ ∪ {ε} and with A ∈ Γ as topmost stack symbol, may read α, change the state to q, pop A, replacing it by pushing α ∈ $\Gamma^*$. The (Σ ∪ {ε}) component of the transition relation is used to formalize that the PDA can either read a letter from the input, or proceed leaving the input untouched.

The computation starts in the initial state $q_0$ with the initial stack symbol Z on the stack, and a string w on the input tape, thus with initial description ($q_0$, w, Z). There are two modes of acceptance. The pushdown automaton either accepts by final state, which means after reading its input the automaton reaches an accepting state (in F), or it accepts by empty stack (ε), which means after reading its input the automaton empties its stack.

**2.5.4 Turing Machines and Undecidability**

A Turing machine (Rich, 2008, p.364) is a theoretical model of a central processing unit (CPU) that handles all data manipulation in a computer, typically using sequential memory to store data. This memory is often represented as an infinitely long tape where the machine can read from and write to.

In formal language theory, a Turing machine can enumerate a subset of valid strings from an alphabet. Such a set of strings is known as a recursively enumerable language. A Turing machine can also be viewed as a model that recognizes valid input strings rather than enumerating output strings.

For a given Turing machine M and string s, it is generally impossible to determine whether M will eventually produce s due to the unsolvability of the halting problem, which highlights a key limitation in computing theory.

A Turing machine can process any unrestricted grammar, meaning it can evaluate first-order logic in infinitely many ways. This capability is notably illustrated by lambda calculus.

Turing machine can be formally defined as a 7-tuple M = (Q, Γ, b, Σ, δ, $q_0$, F) where

- Γ is a finite, non-empty set of tape alphabet symbols;
- b ∈ Γ is the blank symbol (the only symbol allowed to occur on the tape infinitely often at any step during the computation);
- Σ ⊆ Γ \ {b} is the set of input symbols, that is, the set of symbols allowed to appear in the initial tape contents;
- Q is a finite, non-empty set of states;
- $q_0$ ∈ Q is the initial state;
- F ⊆ Q is the set of final states or accepting states. The initial tape contents is said to be accepted by M if it eventually halts in a state from F.
- δ : (Q \ F) x Γ → Q x Γ x {L, R} is a partial function called the transition function, where L is left shift, R is right shift. If δ is not defined on the current state and the current tape symbol, then the machine halts; Intuitively, the transition function specifies the next state transited from the current state, which symbol to overwrite the current symbol pointed by the head, and the next head movement.

An undecidable Turing machine is a Turing machine that cannot solve a decision problem that requires a yes or no answer in a finite amount of time. Conversely, A problem is decidable if we can construct a Turing machine which will halt in a finite amount of time for every input and give answers as ‘yes’ or ‘no’. A decidable problem has an algorithm to determine the answer for a given input.

## 2.6 Summary

We have briefly covered all the concepts we will be needing to conduct the research work needed for this project. We have briefly introduced the mathematical base that is needed to complete this project, along with recent advances in deep learning and generative AI. Post that, we also touched upon ideas from point cloud processing that are going to be essential for this project later on. Lastly, we concluded this literature review by briefly discussing standard results from the theory of computation we will be using.

# CHAPTER 3: RESEARCH METHODOLOGY

## 3.1 Introduction

This work focuses on the formalization of the concept of a neural network into an abstract structure which preserves all the properties in the form of a token, and also preserves all the relationships between those tokens. It is then shown how this formalization can also be implemented in the form of a formal data structure for efficient archival and editing of media data in AI applications.

## 3.2 Algorithms & Techniques

### 3.2.1 Dynamic Graph CNN

DGCNN (Dynamic Graph CNN) (Wang et al., 2019) is a neural network that utilizes small modules known as 'EdgeConv'. In each layer of the network, dynamic graphs are computed. DGCNN's most crucial component, known as EdgeConv, captures the local geometric structure of the point cloud while maintaining permutation invariance. EdgeConv generates edge features that describe the relationship between a point and its neighbors. This module can be implemented in existing 3D point cloud processing networks like PointNet to enhance their performance.

The classification model architecture shown in figure 3.1 takes as input n points, calculates an edge feature set of size k for each point at an EdgeConv layer, and aggregates features within each set to compute EdgeConv responses for corresponding points. The output features of the last EdgeConv layer are aggregated globally to form an 1D global descriptor, which is used to generate classification scores for c classes.

**Point cloud transform block**: The point cloud transform block is designed to align an input point set to a canonical space by applying an estimated $3 \times 3$ matrix. To estimate the $3 \times 3$ matrix, a tensor concatenating the coordinates of each point and the coordinate differences between its k neighboring points is used.

**EdgeConv block**: The EdgeConv block takes as input a tensor of shape $n \times f$ , computes edge features for each point by applying a multi-layer perceptron (mlp) with the number of layer

neurons defined as {a1, a2, ..., an }, and generates a tensor of shape n × an after pooling among neighboring edge features.

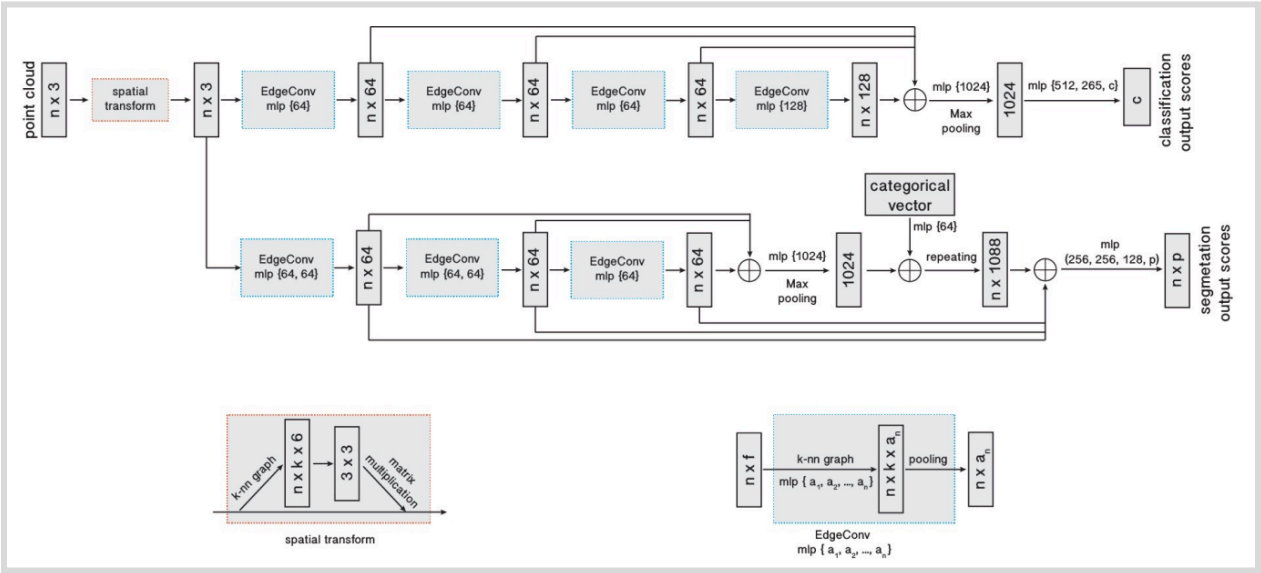


Figure 3.1 : Dynamic graph CNN model architecture (Wang et al., 2019)

### 3.2.2 Vector Neuron based DGCNN (VN-DGCNN)

On the top of DGCNN, we incorporate the concept of vector neuron as defined in (Deng et. al., 2021) to obtain something called as VN-DGCNN that is SO(3) equivariant. It has been tested on classification operations to be permutation-invariant and rotation-invariant. We use the VN-DGCNN variant to perform classification on point cloud streams of data to identify objects of interest.

DGCNN performs a permutation equivariant edge convolution by computing adjacent edge features E'nm followed by a local max pooling:

$$E'_{nm} = \mathrm{ReLU}(\Theta(x_m - x_n) + \Phi x_n)$$ [As defined in (Deng et. al., 2021, eq. 16)]

$$x'_n = \mathrm{Pool}_{m:(n,m)\in E}(E'_{nm})$$ [As defined in (Deng et. al., 2021, eq. 17)]

where $x_n \in R^3$ are per-point features and Θ, Φ are learnable weight matrices.

On the other hand, the VN-DGCNN is defined as follows:

$$E'_{nm} = \text{VN-ReLU}(\Theta(V_m - V_n) + \Phi V_n)$$ [As defined in (Deng et. al., 2021, eq. 18)]

$V'n = VN\text{-}Pool_{m:(n,m)\in E}(E'nm)$ [As defined in (Deng et. al., 2021, eq. 19)]

where $Vn \in R^{C*3}$ are our vector-list representations.

For a detailed explanation of VN-ReLU and VN-Pool, the interested readers are requested to refer to the original paper (Deng et. al., 2021, pp. 5). In principle, VN-DGCNN extends neurons from 1D scalars to 3D vectors and enable a simple mapping of SO(3) actions to latent spaces thereby providing a framework for building equivariance in common neural operations – including linear layers, non-linearities, pooling, and normalizations with a DGCNN backbone.

### 3.2.3 Tree of Clarifications (ToC) framework

Tree of Clarifications (ToC) framework (Kim et. al., 2023) is responsible for breaking down an ambiguous user query (AQ) into small chunks of disambiguated queries, calculating responses to them, and then consolidating the responses from sub-queries to answer the original query entered by the user.

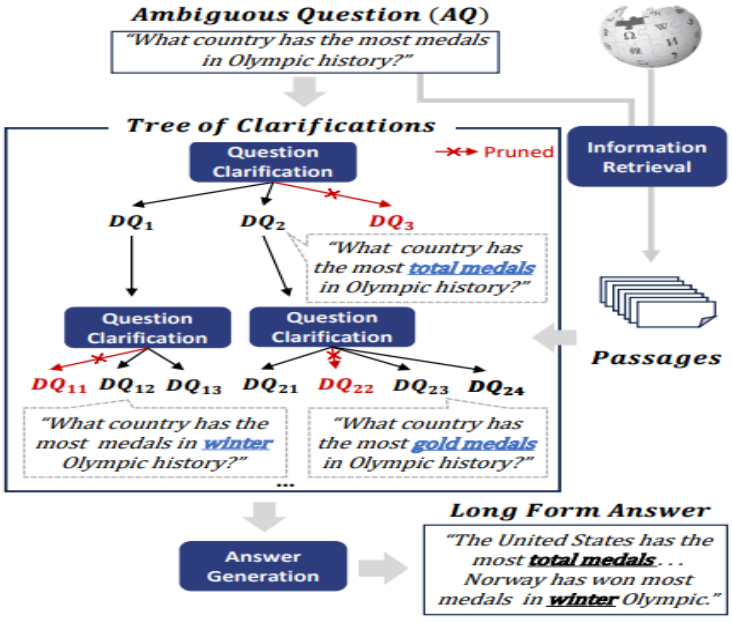
Ambiguous Question (AQ)
"What country has the most medals in Olympic history?"
Tree of Clarifications
Question Clarification
Pruned
Information Retrieval
DQ1
DQ2
DQ3
"What country has the most total medals in Olympic history?"
Question Clarification
Question Clarification
Passages
DQ11 DQ12 DQ13 DQ21 DQ22 DQ23 DQ24
"What country has the most medals in winter Olympic history?"
"What country has the most gold medals in Olympic history?"
Long Form Answer
Answer Generation
"The United States has the most total medals . . . Norway has won most medals in winter Olympic."

Figure 3.2 Tree of Clarifications (ToC) framework (Kim et. al., 2023)

The framework makes use of Retrieval-Augmented Clarification (RAC) technique to first retrieve relevant documents from the attached retrieval systems, re-ranks them and chooses the top-k results. Then it augments them into a single disambiguated prompt. Post this, to effectively

resolve the ambiguous query, it utilizes something called as a recursive tree structure of clarifications.

Starting from the root node with AQ, it progressively inserts child nodes and recursively performs RAC on each of them resulting in a set of disambiguated query-answer pairs. In each expansion step, passages are reranked again regarding the current query, ensuring only the best results move up from the node. This means that we start from a node-perform RAC to obtain best possible resolvable sub-queries or rules, combine or augment them in all possible ways to obtain child nodes, execute them, and merge the results to obtain the parent query result.

### 3.2.4 Church Turing Thesis

In simple terms, Church-Turing thesis (Rich, 2008, p.411) states that every computation that can be carried out in the real world can be effectively performed by a Turing Machine. It is a thesis around the class of computable functions that formally proves the existence of an effective method for calculating a function on the natural numbers if and only if it is computable by a Turing Machine.

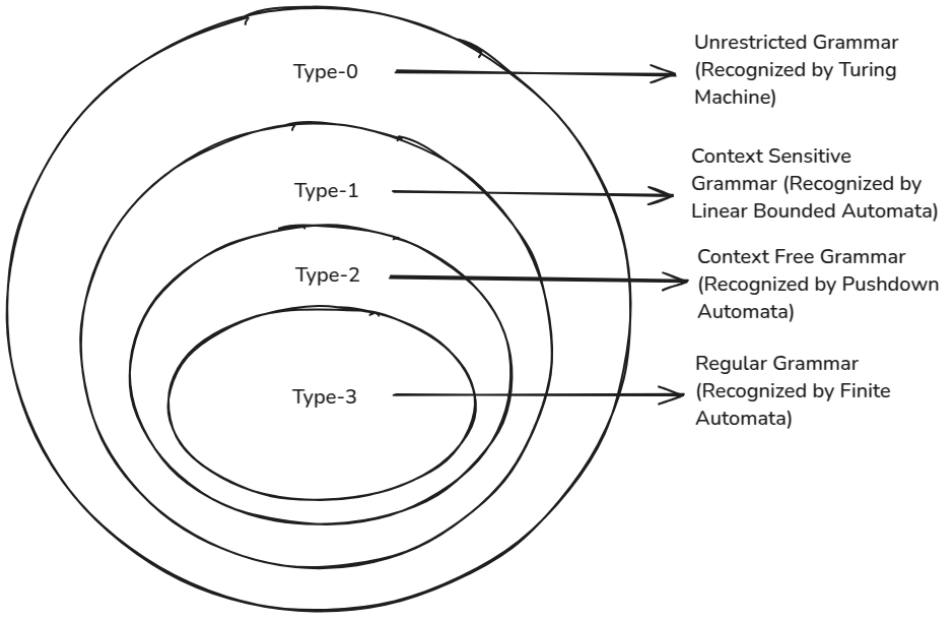


Figure 3.3 Chomsky hierarchy

This result is significant because the language definition or the way of defining what we want to look for in the sequence of classified objects received from our point cloud classifier is expressive enough to cover the class of all computable functions.

As seen in figure 3.3, the class of formal language recognized by a Turing Machine is essentially the most expressive formal language that exists in literature. This means that if there exists a computable function to express what we want to capture in the form of data, there is guaranteed that at-least one language definition must exist for this function. The choice of specification will ultimately depend on multiple aspects such as the complexity of the data stream we are going to model, feasibility of parser implementation, etc.

## 3.3 Methodology

### 3.3.1 End-to-End Pipeline

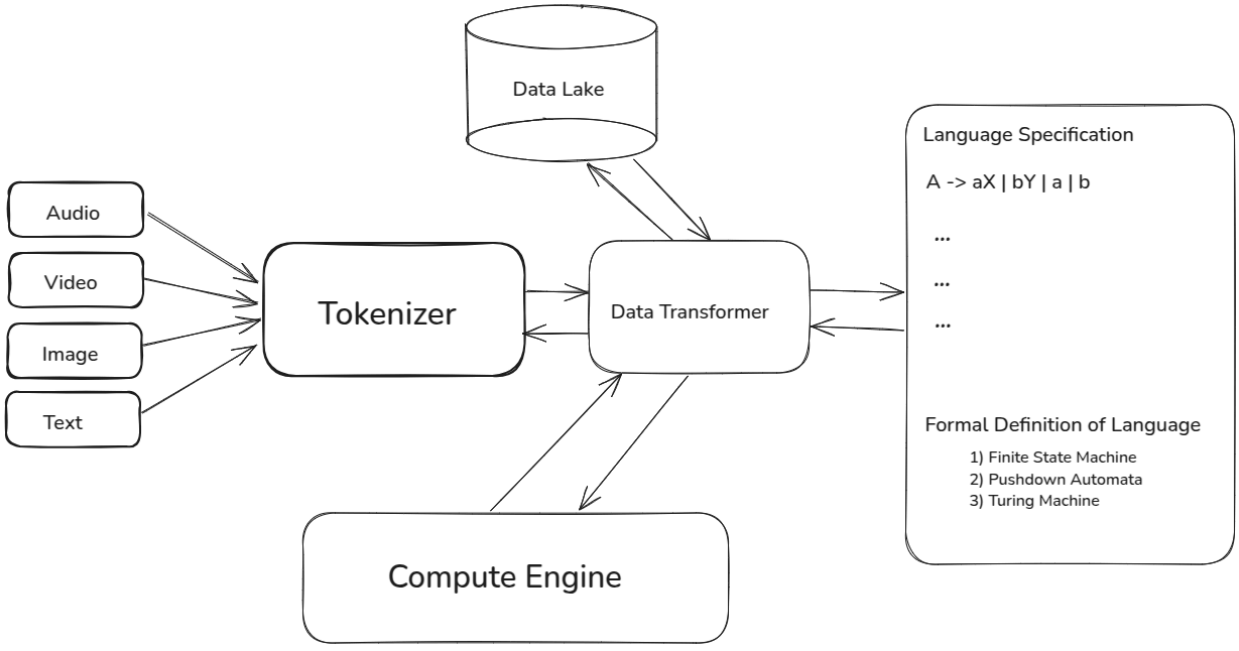


Figure 3.4 End-to-end pipeline

This is the complete end-to-end pipeline for our proposed solution. Up next, we are going to cover every individual component in this architecture.

- **Tokenizer**- This is a specialized module that accepts as input all types of data or as per individual use case, and extracts the discrete stream of point cloud tokens and sends them

to the data transformer for further process. Here, in a general setting, the idea behind tokenization is to break down a stream of binary information into meaningful chunks that can be processed and understood by the software system in a clear and concise manner. Any preprocessing operation such as depth-map calculation will take place here before being sent to the data transformer.

- **Data Transformer**- This module accepts point cloud data streams from the Tokenizer and shapes them into meaningful format with the help of a guided formal language definition. This transformed point cloud data stream is then stored in a standard file format into a data lake consisting of the binary contents of the file as well as a dictionary consisting of the metadata defining the insights captured per the language specification..

- **Compute Engine**- This module is responsible for answering queries as well as performing edits upon the point cloud data stored in the data lake by working in conjunction with the data transformer. It is responsible for accepting and processing the user query, resolving any ambiguity, and coordinating with the data transformer for extracting relevant information from the data lake.

- **Language Specification**- The role of a language specification is to define rules in the specification that needs to be captured and processed by the software system. Here, since we convert all the input types into point cloud data, it is a multi-modal specification in the sense we can express and build rules combining different modalities such as audio, video, text, etc. It also defines any post processing operation that might be needed if a rule is identified in the binary stream, and maintains a many-to-many relationship with the files stored in the data lake in which a given rule was captured or identified.

- **Data Lake**- This is a standard NoSQL Database that stores the raw binary stream in the form of a document record. The core idea of our framework is not to go by a well defined schema to define what we must capture. Instead, look for patterns or desired functional aspects in every data stream instance, and maintain a pointer of every language rule that was found in that file.

### 3.3.2 Data Description

This work is mainly based on two datasets-

- **ModelNet40**: Introduced by (Wu et al., 2015), this dataset contains synthetic object point clouds. It is the most widely used benchmark for point cloud analysis. The dataset is specifically popular because of its various categories, clean shapes, well-constructed dataset, etc.
- **ASQA**: Introduced by (Stelmakh et al., 2022), It is the first long-form question answering dataset that focuses on ambiguous factoid questions. Different from previous long-form answers datasets, each question is annotated with both long-form answers and extractive question-answer pairs, which should be answerable by the generated passage.

### 3.3.3 Data Preparation and Intended Usage

We briefly discuss the datasets that will be referred to in this work and also cover if any preprocessing is going to be needed in order to make them ready for applying into the framework pipeline.

#### 3.3.3.1 ModelNet40

As we will see later, the ModelNet40 dataset described in (Wu et al., 2015) serves the purpose of training the VN-DGCNN classifier used in our framework. It contains 12,311 pre-aligned shapes in the form of CAD files from 40 categories such as chair, sofa, airplane, etc. which are split into 9,843 (80%) for training and 2,468 (20%) for testing.

#### 3.3.3.2 ASQA

As we will see later, the ASQA dataset described in (Stelmakh et al., 2022), is used for training the Tree of Clarifications framework (ToC) used later on in our work. It is the first long-form question answering dataset that focuses on ambiguous factoid questions. It is also used in our framework for resolving the right point cloud tokens from the user query by the compute engine. The dataset contains 5301 long form question-answer pairs that are split into 4353 (82%) for training and 948 (18%) for testing.

### 3.3.4 Implementation

The proposed implementation can be broadly separated into three major steps:

1. **Tokenization** – In this step, we break down a given input stream of media content into a stream of discrete tokens in a standard point cloud representation. These tokens will capture all the information about the individual components of the media content such as pixel information, depth maps, etc.
2. **Archival into an Abstract Form-** In this step, we store the discrete tokens and their underlying relationship in a standard file format containing point cloud data in binary form along with metadata consisting of calculated insights that can be directly worked upon for multiple downstream activities such as prompt answering, image editing, etc.
3. **Query and Edit Archived Data-** In this step, we resolve the user prompt into a set of non-ambiguous queries, execute them and combine their results to return back the required information from the data lake. The same process applies in case of edit operation to identify and perform the required operation on the correct set of information.

In the next few sections, we are going to discuss the different types of operations covered by the proposed framework and how the different elements from our proposed framework coordinate with respect to each other for completing those operations.

#### 3.3.4.1 Transform Data into Point Cloud Representation

We transform the raw data stream into discrete point cloud tokens, and those tokens are correctly stored in a data lake post validation by a data transformer with the help of a formal language specification. We first use VN-DGCNN for point cloud classification and then parse the language specification according to the occurrence sequence of objects received from the classifier. We then store the binary file content along with a dictionary maintaining the traversal path obtained on the language specification as a path of the classification process in a standard file format.

#### 3.3.4.2 Encode Point Cloud Data into Archival Form

The point cloud tokens are validated by the data transformer by referring to the language specification which is a well-defined formal language. Here, if the string sequence containing classified point cloud tokens is completely accepted by the automaton, it is considered as a valid stream and is stored as is in the binary form coupled with the parsed metadata for relevant downstream operations later on.

#### 3.3.4.3 Answering Query upon Encoded Point Cloud Data

We make use of the Tree of Clarifications framework (Kim et. al., 2023) to correctly break down the user query into a set of disambiguated sub-queries, evaluate the response of top k ranked sub-queries, and then summarize the results into one and return it back to the user.

Here, in case when the user asks for some query on the stored data such as “What is the difference between the two buildings in image X”, the compute engine then resolves into the following set of sub-queries: (A) sub-query looking for image X; looks into its metadata dictionary, and (B) resolve the sub-query that stores the exact location of the two buildings; (C) resolve the sub-query to look in the stored depth map information; and (D) then use this information to return back the exact difference between the two buildings back to the user taking into consideration scaling effects.

All of this metadata has already been calculated during the data transformation phase, and the whole operation is working bottom-up and resolving the right set of information.

#### 3.3.4.4 Targeted Editing via Encoded Point Cloud Data

In case of an editing request made by the user, the compute engine, we once again utilize the Tree of Clarifications framework (Kim et. al., 2023) to break down the user query into a set of mini-queries that can be directly resolved to the language specification rules by the data transformer. The data transformer then looks up in all the files stored in the data lake that have calculated or resolved those rules. This metadata is used to accurately resolve the correct binary file and then with the help of an appropriate targeting editing backbone such as (Gu et al. 2024; Zhu et al. 2024), the editing operation is performed and the result is returned back to the user via the compute engine or stored back in the data lake as required.

### 3.3.5 Design Considerations

Here, we are now going to discuss some key design considerations that went into the framework we have just discussed and how it can be amended and applied in a practical setting. We will also discuss how this framework potentially serves the purpose of a general design guideline for building modern AI enabled software systems that are semi-autonomous and functional reactive in nature.

1. Selection of Point Cloud Representation- One of the design choices that the reader must have noticed is why we have opted for point cloud representation for perceiving our data. The reason is straightforward- Based on current research trends, we have a good amount of research backing for efficiently converting all modalities such as Audio, Video, Images, etc into point cloud representation, including a good amount of datasets and foundation models.
2. Choice of Vector Neuron based DGCNN- Another key design choice was using Vector Neuron based DGCNN (Deng et. al., 2021) for classification tasks on point cloud tokens using vector neuron network. This is something that was based on its performance superiority over other counterparts as we will see later in chapter-5 .
3. Choice of Query Resolution Approach for the Compute Engine- Another key design we have discussed is the Tree of Clarifications (ToC) framework (Kim et. al., 2023) for resolving ambiguous user queries into a set of non-ambiguous queries and then resolve and return the relevant response. This was again, a design choice based on recent research trends and the general guideline here is that as a thumb rule, the implementor must identify and incorporate the most viable query resolution mechanism.

The core ideology will always remain the same- depending on the individual project needs, the implementor must identify the right representation for tokenization depending on factors such as available DevOps service, ease of training, dataset availability, etc. Secondly, the implementor must also identify the right classifier for those token streams depending on training, deployment and long term maintenance costs.

A direct implication of the mentioned design choices is that we intend to groom and evolve this framework as a standard template for building AI enabled software systems. To understand this

better, consider a simple full stack application being built for a departmental store. The store management hired a reputed software development agency, briefed them about the functional requirements and got the application built for them as expected in a matter of a few months. The Tech Stack consisted of a traditional backend in (let's say) Java Spring Boot, a PL-SQL Database, and a React JS Frontend.

Here, as we know, businesses are rarely static with ever shifting market dynamics. Let's say the management wants to tinker with some new ideas such as Gold Membership offering extra discounts for certain customer segments, a new line of product offerings and quick commerce. A standard development approach will require adding backend logic, new schema objects as well as logic to incorporate those schema all the way up into frontend and backend.

Alternatively, with our discussed framework, the schema would essentially store everything in the form of an object or a binary document, and the task of adding a new product catalogue would essentially reduce into a task of fine tuning the classifier with new labelled data. The backend and frontend code would essentially remain untouched thereby massively reducing development costs.

Similarly, adding a new line of business or personna such as Gold Membership or quick commerce would essentially involve defining new business rules in the language specification to identify those relevant peronnas and linking relevant java classes to those specification rules such as additional discount apply, delivery prioritization, etc. All of this comes under configuration and is something the management can tinker around on their own using a generic frontend interface as we will see later in chapter-4.

### 3.3.6 Evaluation

As we will do further performance related evaluation later in chapter-5, the proposed framework qualifies to be used for both academic as well as industrial purposes in standard AI enabled software system development. It can be used for both rapid prototyping as well as formal verification of software systems wherein a user can directly operate on unstructured data without the need to convert it into standard tables or database forms. It captures the data stream in a consistent and well defined standard file representation that is well understood because of the

calculated metadata stored in it. The Tree of Clarifications (ToC) based query resolving mechanism as we will see later in chapter-5, also outperforms the other similar baseline models and hence its usage in our framework is also justified.

## 3.4 Tools

### 3.4.1 Software

The software requirements that have been used for this research work are as follows:

- Salesforce.com Developer Account.
- Internet Browser such as Google Chrome.
- SLDS-CLI for Salesforce Console Integration.
- Lightning Web Components (Frontend Framework).
- VS Code for Development.

### 3.4.2 Hardware

The hardware specification that has been used for conducting this research work are as follows:

- A desktop workstation with 13th Gen Core i7 Hexa Core processor, 1TB SSD and 16GB ram.
- A working internet connection with 100 mbps speed.
- A RTX-3060 6GB GPU for training and evaluating various models.

## 3.5 Summary

In this chapter, we discussed the research methodology that is being used for this project. We briefly introduced the datasets we will be using, followed by the data transformation and preprocessing operations that will be done. Post that, we discussed the end to end pipeline and architecture for our proposed solution, explaining all the modules in detail. We then concluded this chapter briefly listing down the software as well as the hardware requirements needed for the project.

# CHAPTER 4: IMPLEMENTATION & SIMULATION

## 4.1 Introduction

In this chapter, we will be discussing implementation details for all the individual building blocks of our solution, including data preparation specific details. The purpose of this chapter is to provide low level inner details of all the components that we have proposed earlier.

## 4.2 Data Preparation

As mentioned earlier, our solution is capable of accepting all forms of input data stream such as Text, Audio, Video, and Image data. This makes it versatile for a wide range of applications such as Speech Processing Engine, Transcription, Captioning, Reasoning, etc. as we see later. For the purpose of this project, we have opted for the dataset ModelNet40 as mentioned earlier, which is a collection of 12,311 pre-aligned shapes in the form of CAD files from 40 categories such as chair, sofa, airplane, etc. We can use a managed api for breaking down these streams into a set of individual frames or images that can be passed down into a tokenizer.

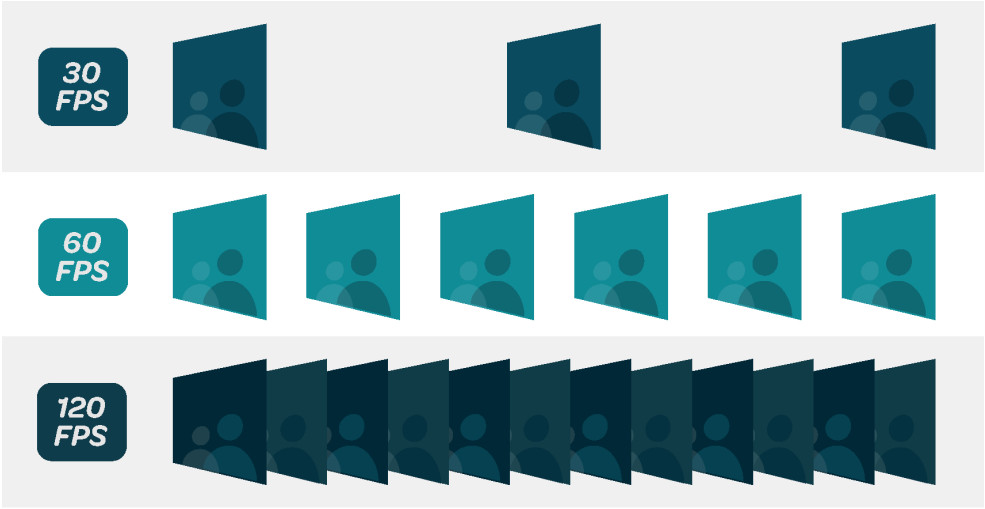


Figure 4.1 Frames per second (illustration) (Klapheke, D., 2023)

It is important to note that, in a typical production scenario, we refer to this as Frames per seconds (FPS), and this process can be automated with MLOps Solutions provided by all major cloud providers such as AWS, GCP, Azure, etc.

## 4.3 Tokenizer

The image frames are tokenized in the form of point clouds using a 3D data processing library such as “Open3D” in python to convert the RGBD (RGB with Depth Information) data from image frames into point clouds. Here, for depth estimation, we can use DepthAnything v2 (Yang et al., 2024) for 2D Image to 3D point cloud data.

For practical feasibility, certain variants are available such as Voxels, Octree, etc as shown in figure 4.2.

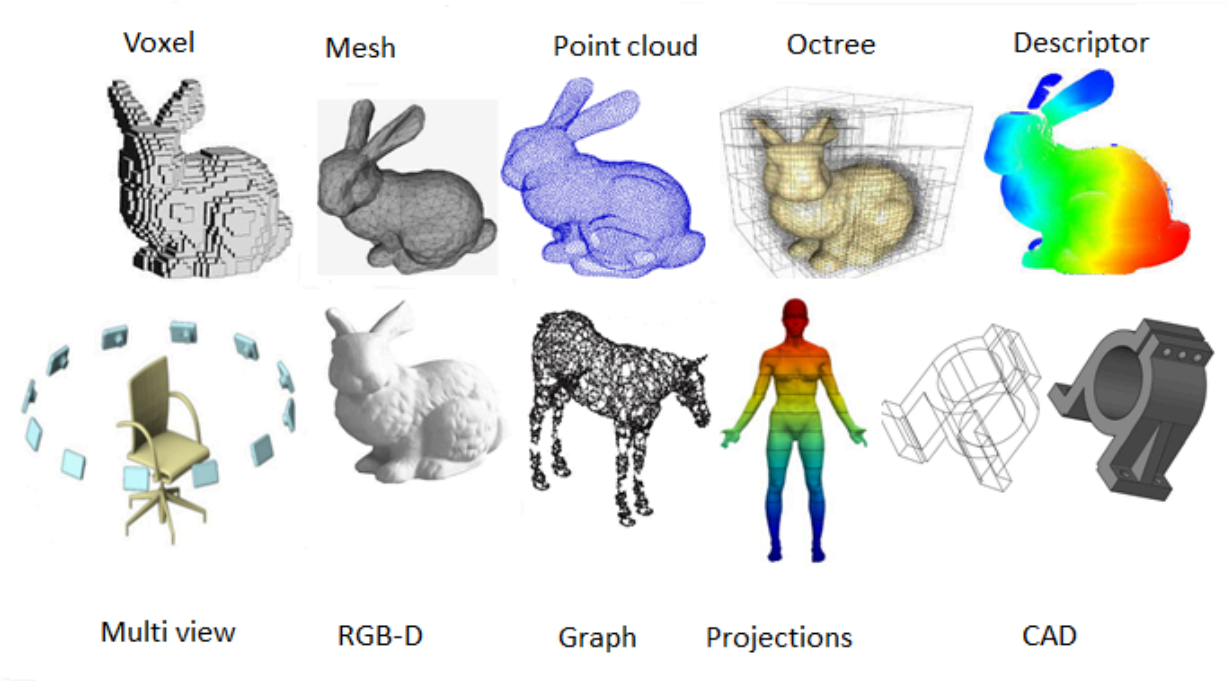


Figure 4.2 Point cloud and other variants (illustration) (Gezawa A.S., et al., 2020)

## 4.4 Data Transformer

The point cloud data is then classified using a data transformer which is basically a implementation of VN-DGCNN that, in conjunction with guidance from the formal language specification, captures the path taken by the grammar while parsing the classification sequence, and records this path in the data lake in the form of a metadata dictionary. While doing so, It also executes any associated semantic action with the particular grammar definition which could be a custom non-linear path, or a post-processing cleanup operation. It is responsible for parsing as well as validating the point cloud token stream as it is received.

As can be seen next in the figure 4.3, the transformer takes up as input every point cloud token one by one, and continues to parse the stream until it is fully read and no error has occurred. In this state, it is considered to be a valid stream and the input is accepted. Else, the input is discarded. This design also ensures strong guardrails in practical applications around the data.

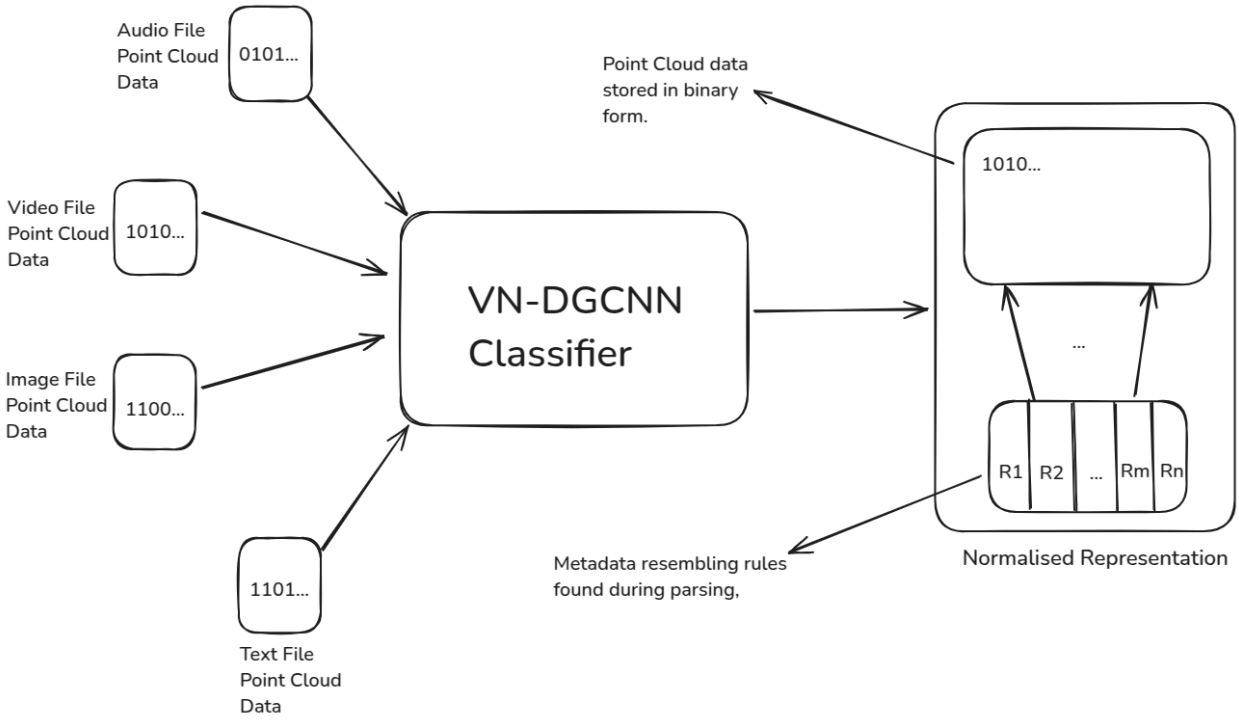


Figure 4.3 Data transformer implementation

Here, the data transformer is responsible for validating inputs while parsing point cloud token stream as well as performing guided edits on the token stream when needed to do so. It also uses the language specification to correctly parse and reason on the user queries related to the data stored in the lake. In this work, we have used the Tree of Clarifications (ToC) (Kim et. al., 2023) which is a framework for recursively constructing a tree of disambiguation for the Ambiguous Query (AQ) via few-shot prompting leveraging external knowledge and using it to generate long-form answers through the relevant metadata. The data transformer then uses this information to return back the stored point cloud tokens from the data lake.

To summarize, the data transformer first validates and classifies the stream of point cloud tokens using an implementation of VN-DGCNN and generates additional metadata to maintain insights captured in the binary data. It then stores this in a normalised form in the form of a standard file

format consisting of this metadata and the binary file content in the data lake and also executes any semantic action associated with the captured insight defined in the language specification.

## 4.5 Formal Language Specification

Formal Language Specification is a well defined language definition as shown in figure 4.4 that is used by the data transformer to validate incoming stream of tokens and store the related metadata into the data lake. It is also referred to in case there is the need of an inference or targeted editing in the stored binary data that has to be done, such as “Tell me more about the Health Vitals while referring to the X-ray image point cloud stored in the data lake.” or “Given the image of a boy with red t-shirt, change the T-shirt color to blue”. It is, by design, strong enough to capture all types of business rules or behaviors that need to be captured as a direct result of the fact that it covers the class of all computable functions per Church Turing thesis (Rich, 2008, p.411).

Figure 4.4 Formal language definition schema

The idea is to standardize all input stream types such as text, audio, video, speech, etc into a common format or a modality such as point cloud token data and specifically look for rules such as “Image of a boy that looks like this <point cloud data of boy’s face> with long hair.” It could be a case of twins who only differ in hair cut as an example.

## 4.6 Compute Engine

The compute engine is responsible for holding the logic related to editing the stored point cloud information in the data lake as well as answering queries related to the stored point cloud data. It does so with the help of a bottom-up approach using the Tree of Clarifications (ToC) framework. The engine first resolves the user query into disambiguated sub-queries that are actually insights calculated by the language specification. This is essentially breaking a large computable function into a set of mini functions that the system knows how to calculate. Then, the engine uses approaches such as collating a list of files for which those mini functions were calculated, applies any filter resolved, then combines the result of all those mini functions to return the result or the insights requested by the user.

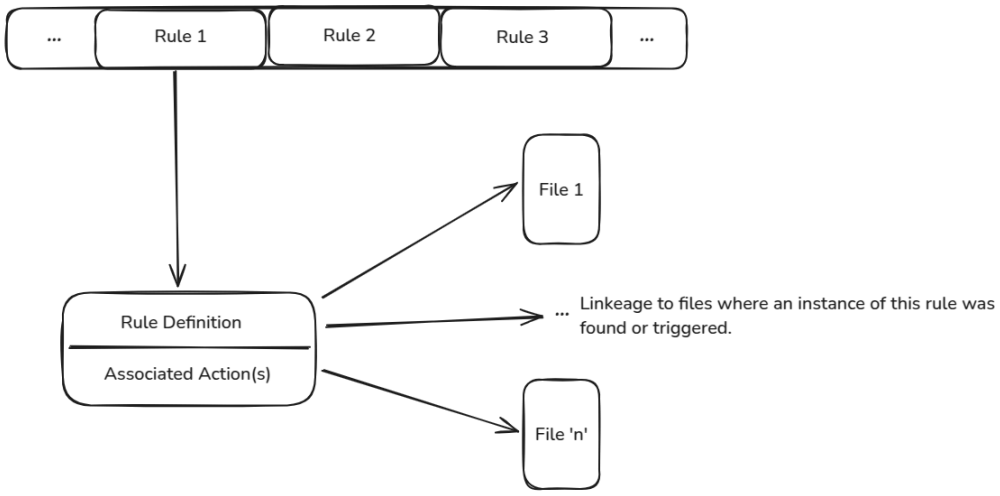


Figure 4.5 File directory structure

As shown in figure 4.5, this linkage for a computed response rule and the relevant binary file in the data lake is maintained in the form of a many-to-many relationship with the language specification rule which is referred to by the compute engine with the help of data transformer.

## 4.7 Sample Implementation

Here, we are going to cover an end-to-end implementation example of the framework and techniques in general that were discussed in this work. The objective is to give a complete walkthrough of the solution that we have built and also give the reader an opportunity to explore and reproduce it on their own in the form of a practical implementation.

### 4.7.1 Demo URL

Before we proceed, the reader is encouraged to investigate and evaluate the source code for the simulation shared in the project Github directory and explore it at their own end.

Github URL: https://github.com/sanketa124/tobyapp

The steps needed to reproduce this work have been clearly outlined in the readme file within this directory itself including a thorough description of the directory structure.

It also includes the sample data that has been used in the following simulation along with all the metadata for the configuration steps done from our end such as reports and dashboards setup, schema setup, permissions setup, etc.

### 4.7.2 Main Screen

Now, once logged in, the user will land onto the home screen shown in figure 4.6:

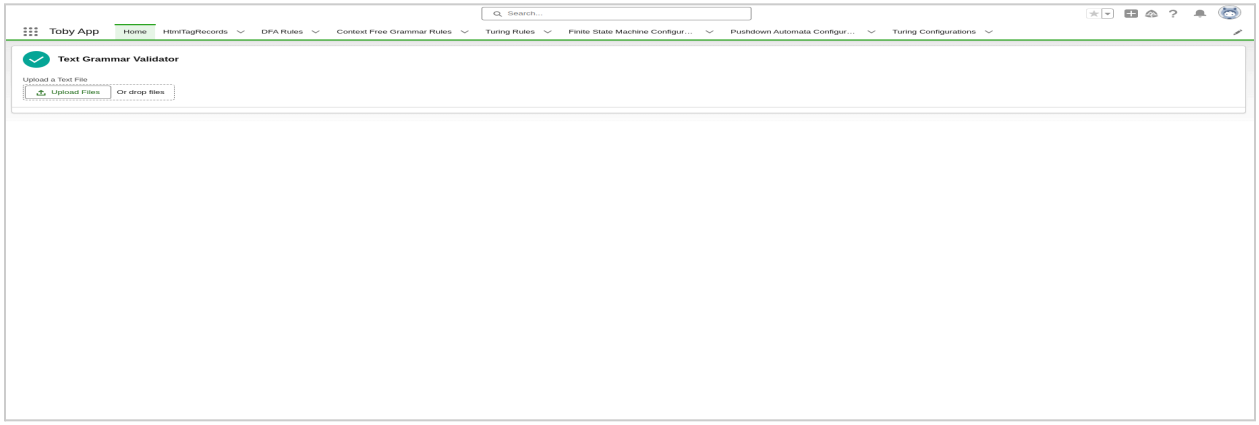


Figure 4.6: Home page screen

Here, the reader needs to upload the “sample.html” file downloaded earlier and upload it via the Upload Files button shown above. Once done, you will get the screen shown in figure 4.7.

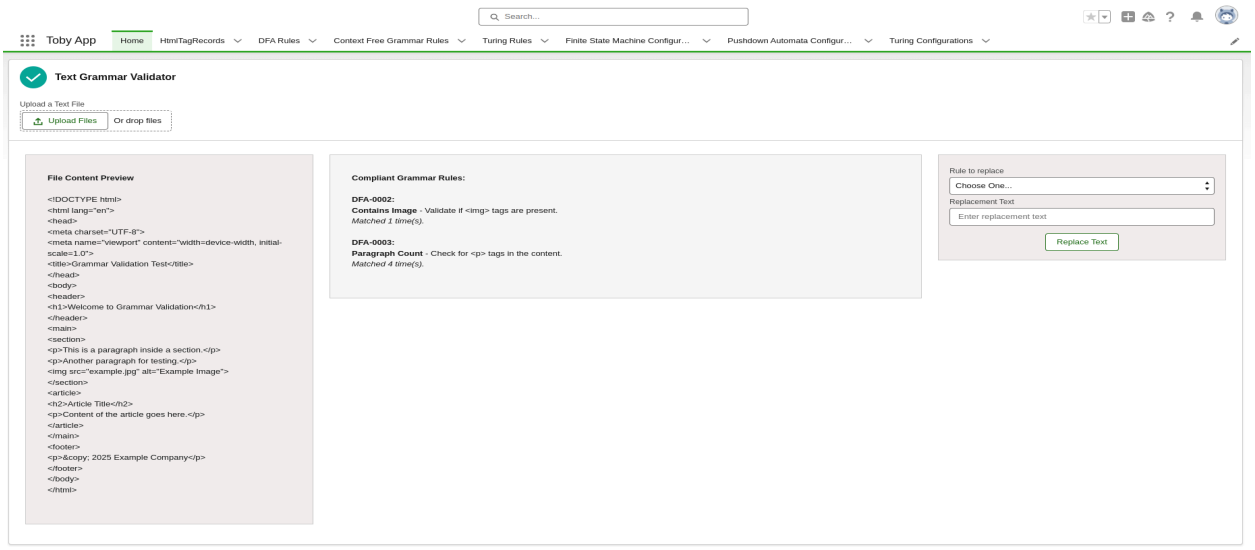


Figure 4.7: Sample upload of HTML file

To explain what we have done, we simulated a real world scenario wherein we will be ingesting a text stream of data (HTML file in this case) which is matched against a set of rules that have been configured in the system. Out of those ‘n’ rules, we may have a hit on, let’s say ‘m’ rules in the data stream which will be shown to the user as above.

Also, on each rule, we have also defined an email alert for this implementation which is received as shown in figure 4.8 and figure 4.9.

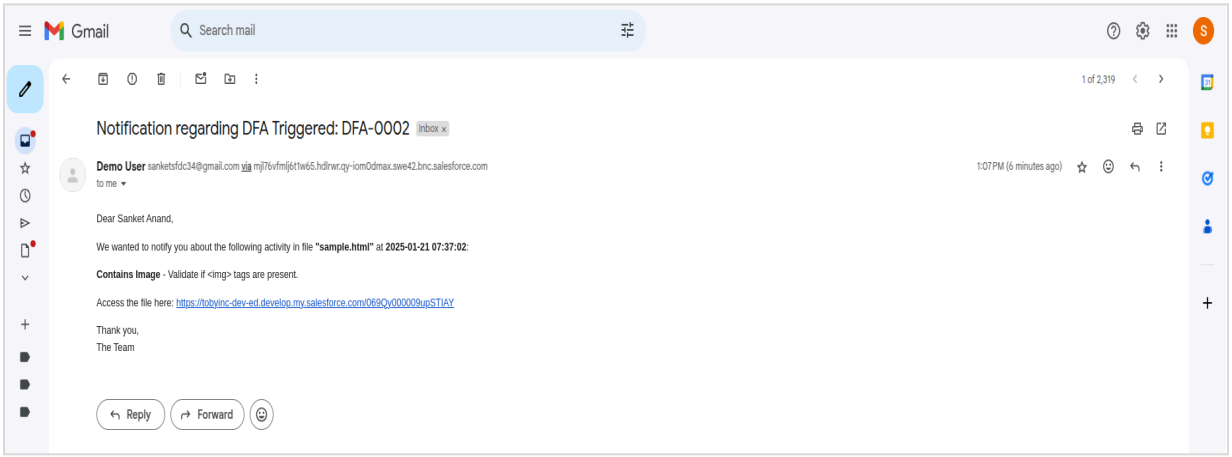


Figure 4.8: Email alert for DFA rule hit

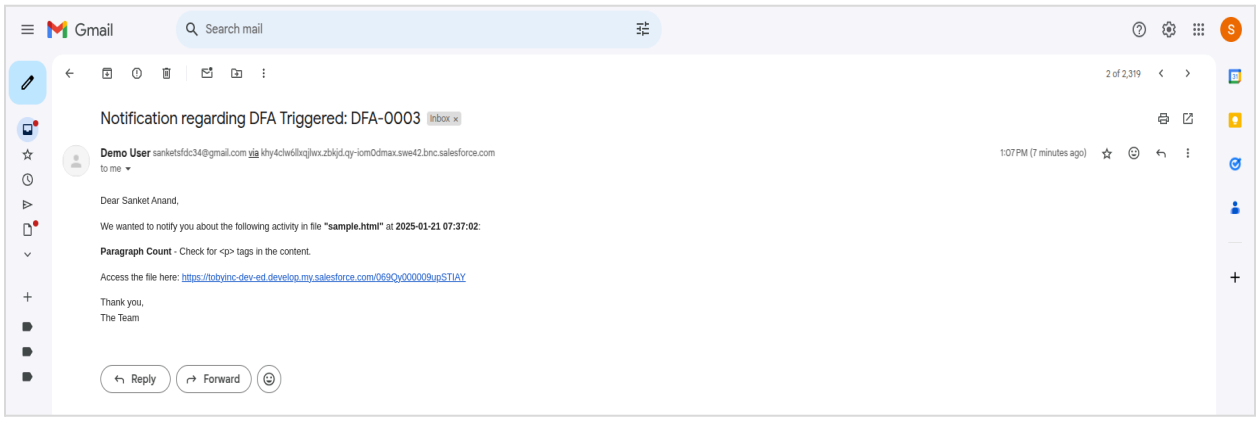


Figure 4.9: Email alert for DFA Rule hit (contd.)

This is just one example, in a general setting, this type of behavior can be extended to virtually anything in practice like a java method, or some secondary code that must be executed if a particular rule is hit. This is particularly useful in alerting systems such as autonomous CCTV security systems, etc.

Now, we look back at the main screen shown earlier and look at the dropdown in the right-most column and select “DFA-003” from the dropdown. Then, we enter a replacement text in the text box below “<test>” and click on the “Replace Text” button as shown in figure 4.10.

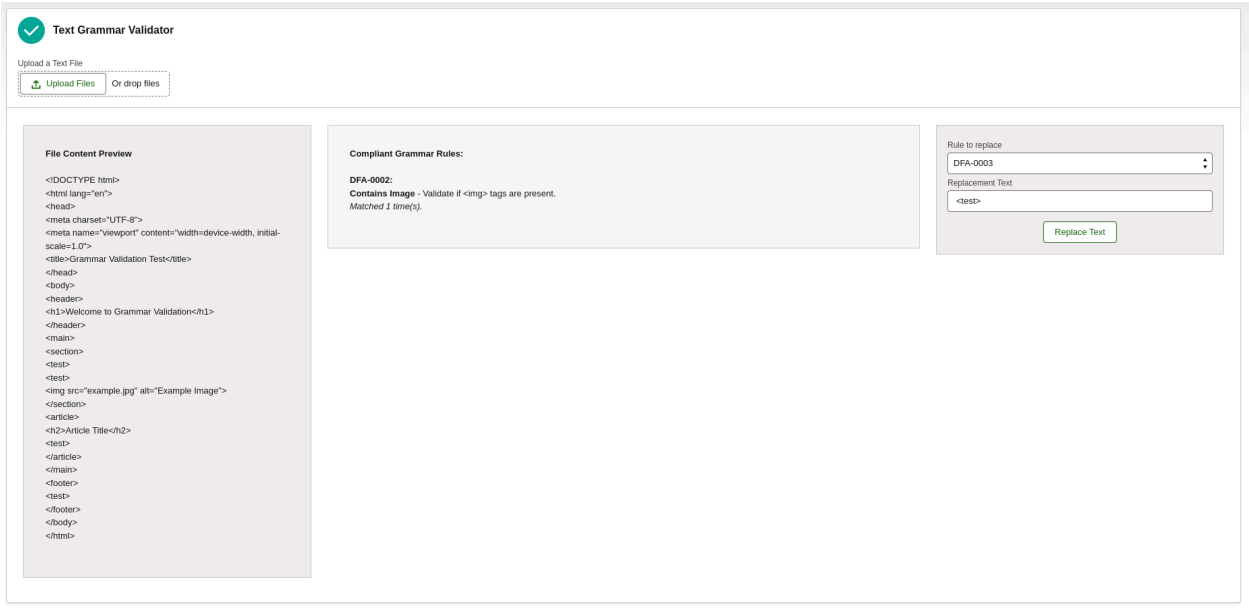


Figure 4.10: Editing of metadata using “Replace Text”

As can be seen in the left-most column, all the <p> tags have been replaced with the <test> tag as shown in figure 4.10. The idea is to capture data in the form of a standardized token stream, and use a certain set of business rules to identify and perform localized edits in the captured data.

Sample Image provided to the API:

Figure 4.11: Sample image upload to OCR API (Emergency sign, n.d.)

Very similarly, we call a Standard OCR API that takes as input an image URL and provides detected objects with bounding box coordinates in a JSON format.

The URL is:

```
curl -X POST -H "Authorization: Bearer <TOKEN>" -F
sampleLocation="https://www.publicdomainpictures.net/pictures/240000/velka
/emergency-evacuation-route-signpost.jpg" -F task="text" -F
modelId="OCRModel" https://api.einstein.ai/v2/vision/ocr
```

Here, the sampleLocation is the URL to the jpg file shown in figure 4.11. For more information on the Einstein OCR API, please refer:

https://help.salesforce.com/s/articleView?id=release-notes.rn_einstein_vision_ocr_ga.htm&release=226&type=5

We would like to, however, highlight that this solution is not vendor locked in any way and in practice, any OCR or speech to text managed service can be used.

Sample Response from the API:


```
{
  "task": "text",
  "probabilities": [
    {
      "probability": 0.99937266,
      "label": "ROUTE",
      "boundingBox": {
        "minX": 582,
        "minY": 685,
        "maxX": 1151,
        "maxY": 815
      }
    },
    {
      "probability": 0.99471515,
      "label": "EMERGENCY",
      "boundingBox": {
        "minX": 361,
        "minY": 208,
        "maxX": 1383,
        "maxY": 346
      }
    },
    {
      "probability": 0.99469215,
      "label": "EVACUATION",
```

```
      "boundingBox": {
        "minX": 331,
        "minY": 438,
        "maxX": 1401,
        "maxY": 570
      }
    }
  ],
  "object": "predictresponse"
}
```


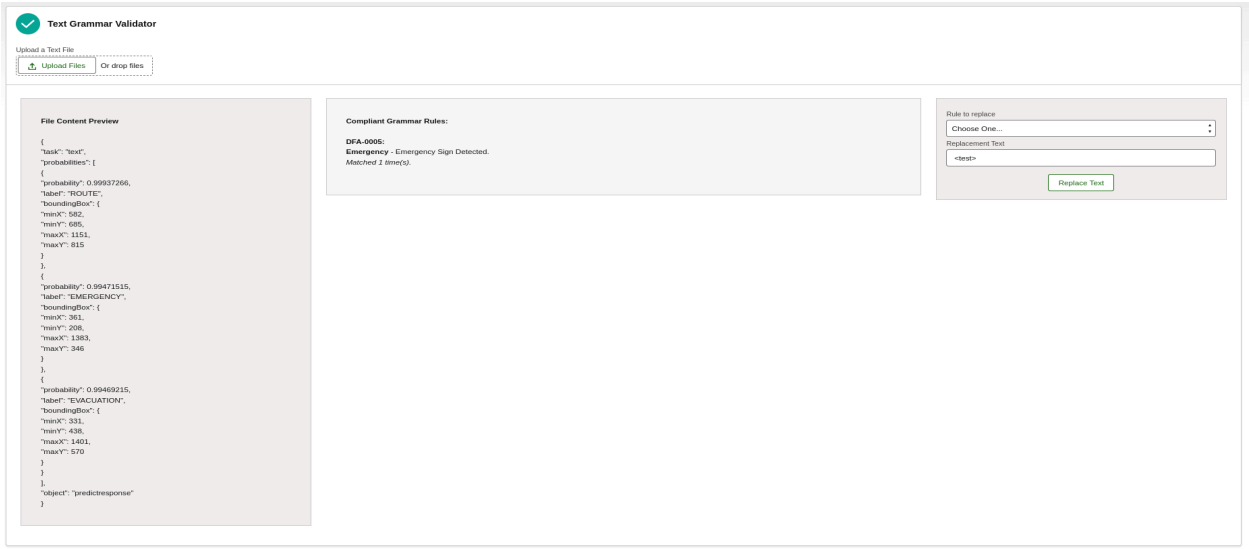


Figure 4.12: JSON upload for OCR API response

The reader can download this JSON file directly from the Github directory as already mentioned earlier. Then, we upload this “sample.json” file on the initial screen as shown in figure 4.12.

Here, as shown above, the business rule designed to capture any “Emergency Sign” has triggered which is an expected behavior. Also, we can perform an edit operation as shown in figure 4.13 very similarly by selecting “DFA-005” in the “Rule to replace” dropdown and entering the replacement text.

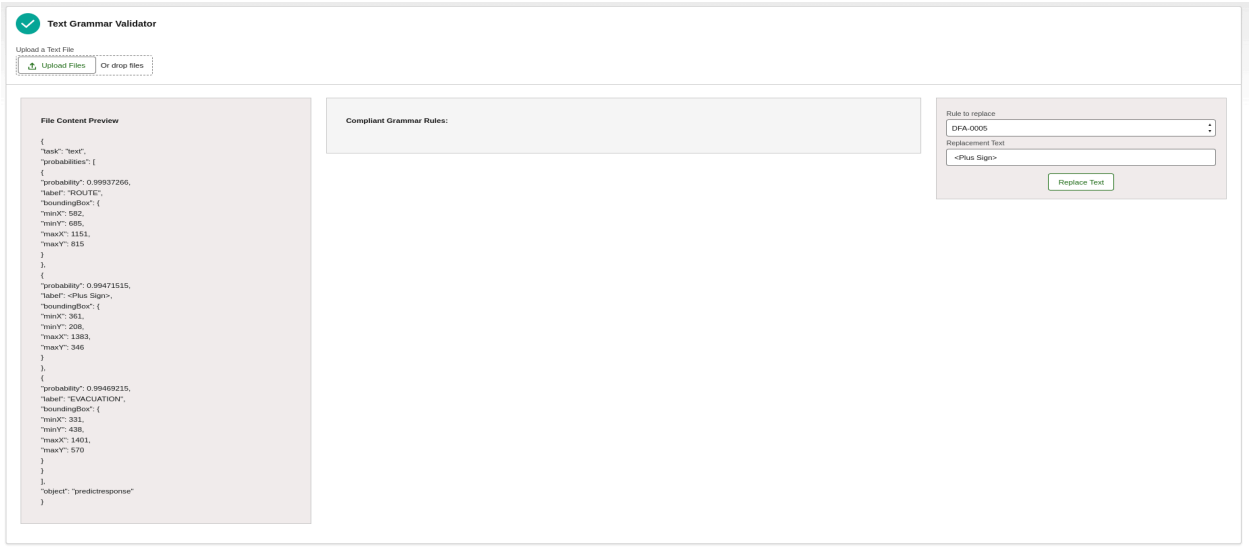


Figure 4.13: JSON replace for OCR API response

In practice, this can then be sent as a prompt to any localized editing solution in the form of an api callout to perform a localized editing within that specific bounding box using a managed service.

### 4.7.3 Creating a New Rule

Now, we are going to demonstrate to the reader how to test out this implementation with your own rule. In order to proceed, the user must click on the DFA rule tab and click on the “New” button as shown in figure 4.14. Here, in the associated DFA, the user can select “FSM-001” and in the alert owner, we have selected “Demo User” as the one who should be receiving email alerts upon a rule match in the data stream. In the Regex Expression, the user can enter the Regex expression needed to be matched.

As already discussed, this framework is designed to be autonomous in nature in the sense there is no need to configure or spend time on schema design and configuration. Instead, it is driven by a grammar, very similar to how a compiler works.

Here, we want to highlight that this is not a full fledged implementation like which finite state machine to match against, or backend logic needed to match against context free grammar rules or turing rules as of now, but the idea will remain essentially the same- depending on the complexity of rule that needs to be defined.

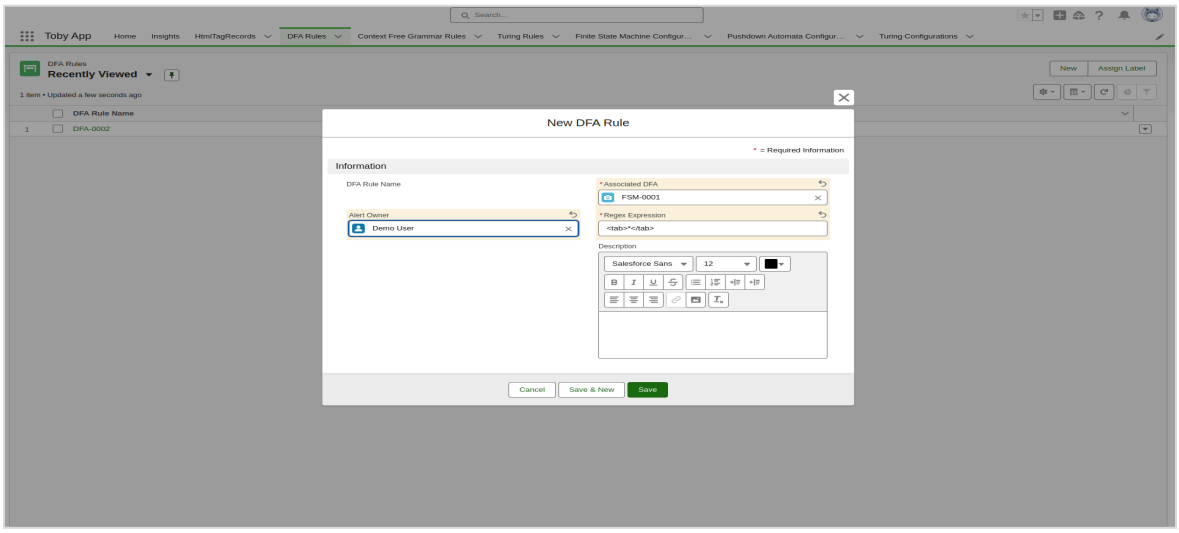


Figure 4.14: Create new DFA rule screen

Finally, we click on the “Save” button and the rule is ready to be used next time a file is uploaded as shown in figure 4.15.

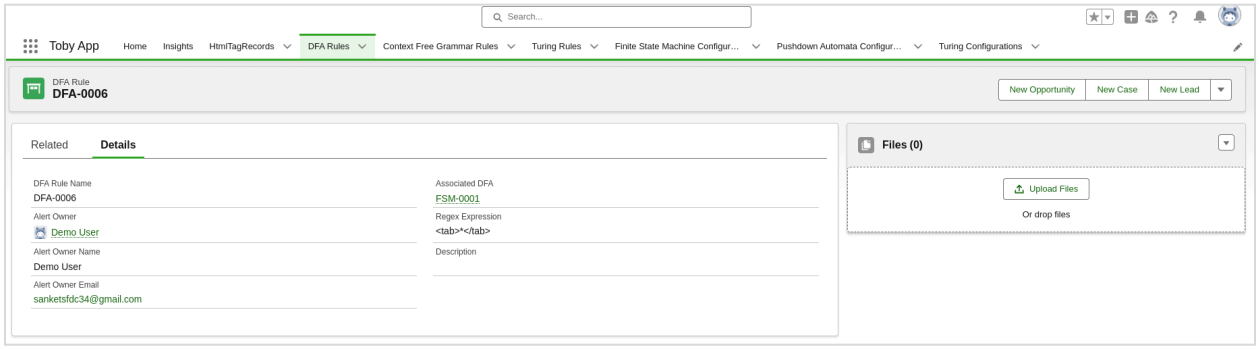


Figure 4.15: DFA rule created detail page

Here, for evaluation purposes, the reader can click on the “Demo User” hyperlink shown above. This will take them to the user detail page as shown in figure 4.16.

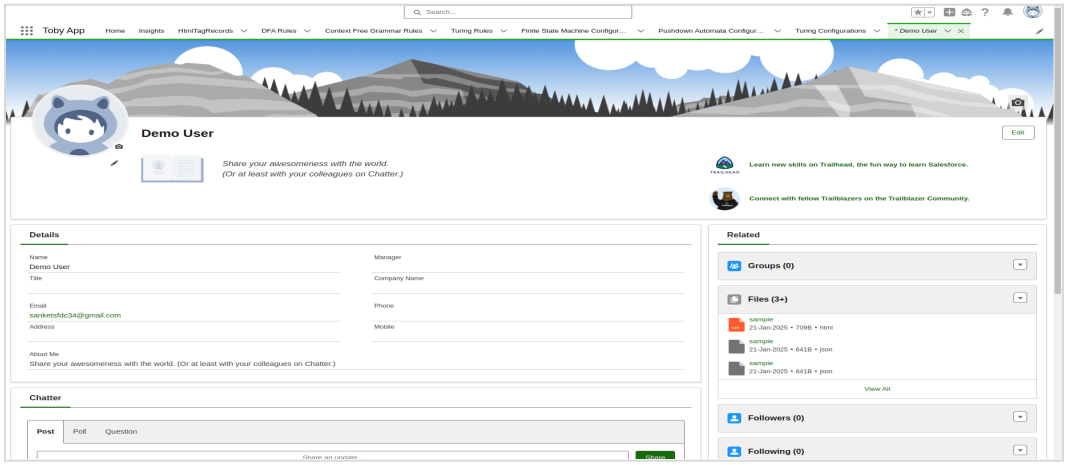


Figure 4.16: User detail page

Here, the reader can click on the "Edit" button in the top right and update the email id as shown below with their own to test the email alert functionality as shown in figure 4.17.

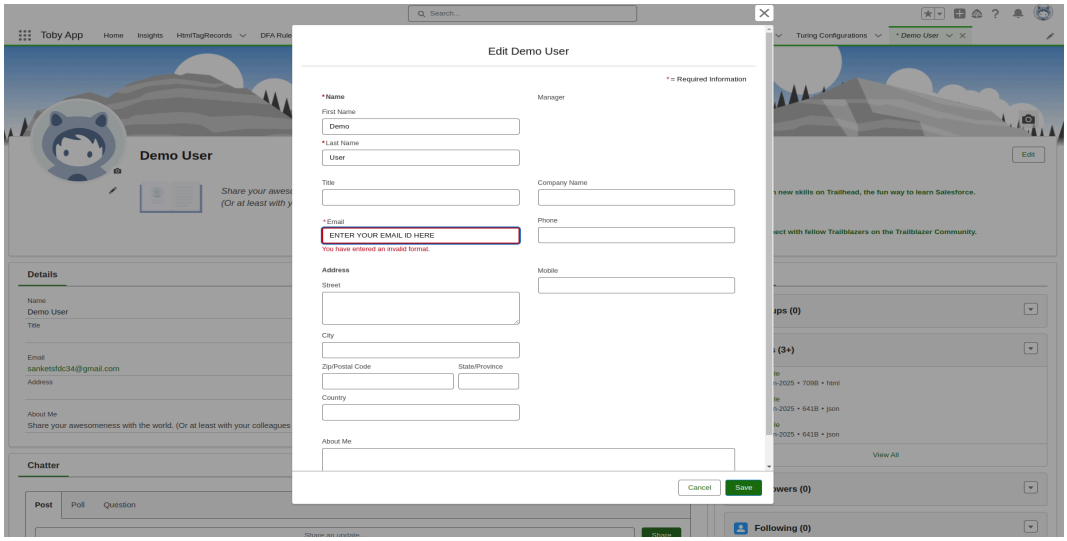


Figure 4.17: User edit screen

Post that, the reader can now navigate back using the "Home" tab and test out the new rule in action that has just been created.

### 4.7.4 Reporting and Alerts

An additional feature that we want to highlight is the ability to report on rules triggered. As a manager, we would be interested in analyzing which all rules were triggered in my data traffic in a typical use case. In that case, we can click on the "Insights" tab and have a look at the donut chart below showing DFAs triggered grouped by types of triggers as shown in figure 4.18.

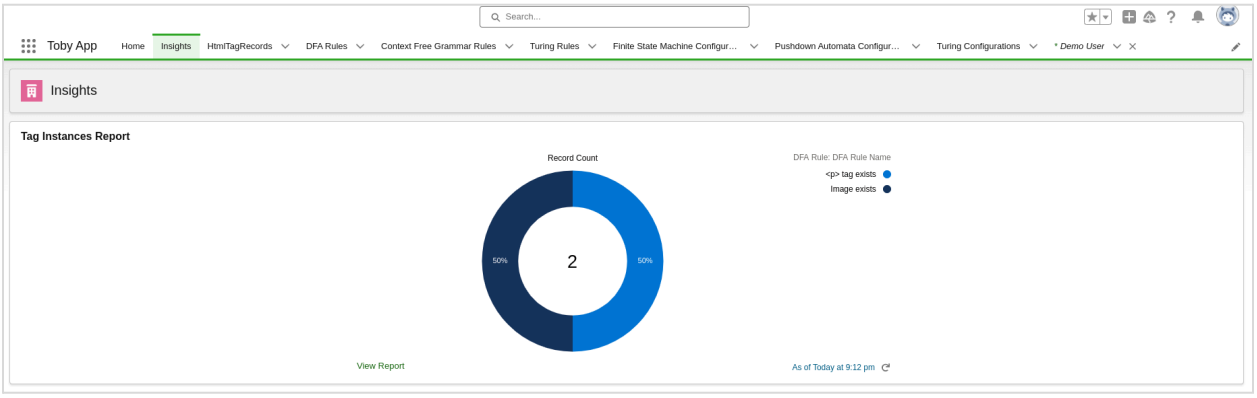


Figure 4.18: Insights screen

Here, clicking on the “View Report” link in the left bottom opens up the report detailed view shown in figure 4.19.

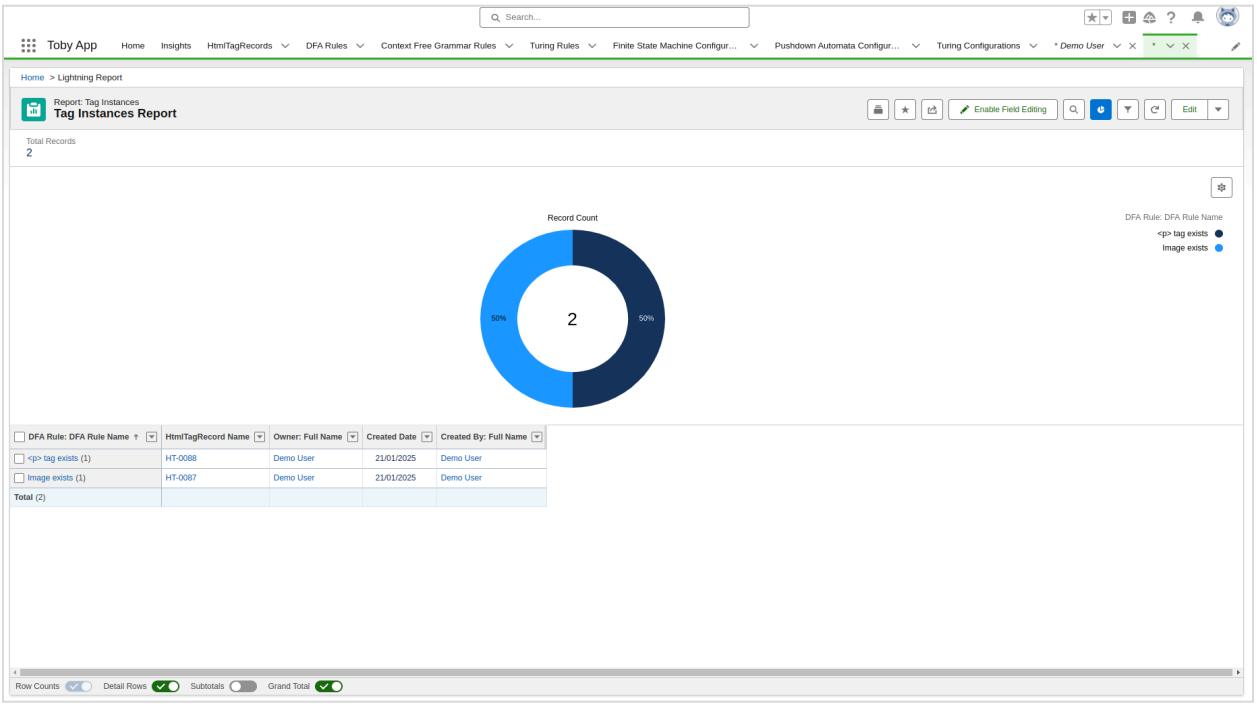


Figure 4.19: Report detail screen

Here, again we want to call out that although for the purpose of this walkthrough we have used certain managed services such as the OCR API to speed up the overall simulation process, our ultimate vision is to release this research work in the form of a open source SaaS Application at some point in near-future using the most suited tech stack such as react for the frontend and django for the backend microservice.

### 4.7.5 Final Word

The reader is encouraged to set up and explore the implementation demo in their own live sandbox through the steps shared in the source code directory. The code and the configuration itself is self explanatory and has been written in such a way that it is relatively straightforward for the reader to follow through and reproduce the main ideas we have been aiming to communicate in their preferred programming language or follow-up implementations.

## 4.8 Summary

In this section, we discussed the implementation of low level details for our proposed architecture and covered all the individual building blocks in great detail. As demonstrated in the chapter, the proposed framework is highly generic in nature and is capable of working with a wide range of input types as well as highly customizable in terms of how the individual building blocks are built and structured.

# CHAPTER 5: RESULTS & EVALUATION

## 5.1 Introduction

This chapter looks into the correctness of the discussed framework and its relevance to the stated objectives. The investigation is done in both qualitative and quantitative manner:

- Quantitative Evaluation: In section 5.2, we investigate the performance evaluation of the building blocks that we have incorporated in our proposed architecture on standard evaluation benchmarks. This will give us a good impression around the kind of performance we can expect out of this work when it is applied to real world applications.
- Qualitative Evaluation: In section 5.3, we do a qualitative evaluation and discuss the flexibility of our framework and evaluate what kind of versatility we can expect out of it in real life use-cases.

## 5.2 Quantitative Evaluation

In this section, we discuss the performance aspects of our proposed framework by looking into the performance benchmarks of all its major individual building blocks since these directly impact the overall performance of the solution.

First, the DGCNN (Wang et. al., 2019) responsible for processing point cloud streams and classifying them has been benchmarked on the ModelNet40 dataset as shown below. Here, as can be clearly seen, DGCNN clearly outperforms the other point cloud classification solutions available. Interested readers are encouraged to refer to the original paper for a more detailed performance benchmarking study.

Table 5.1: Classification comparison on ModelNet40 dataset (Wang et. al., 2019)

| Model Name | Mean Class Accuracy (%) | Overall Accuracy (%) |
|---|---|---|
| 3D ShapeNets [WU ET AL. 2015] | 77.3 | 84.9 |
| VRN (Single View) [BROCK ET AL. 2016] | 88.98 | - |
| VRN (Multiple View) | 91.33 | - |

| | | |
|---|---|---|
| [BROCK ET AL. 2016] | | |
| PointNet [QI ET AL. 2017B] | 86.0 | 89.2 |
| PointNET++ [QI ET AL. 2017C] | - | 90.7 |
| PointCNN [LI ET AL. 2018A] | 88.1 | 92.2 |
| PCNN [ATMZON ET AL. 2018] | - | 92.3 |
| DGCNN | 90.2 | 92.9 |

Next, in the compute engine, the Tree of Clarifications technique implemented for open-domain question answering (ODQA) can be clearly seen to outperform other alternatives in the long-form QA category on the ASQA dataset (Stelmakh, 2022). Following (Stelmakh et al. 2022), the benchmarking is done on 3 parameters: Disambig-F1 (D-F1) which measures the factual correctness of generated predictions. It extracts short answers to each DQ and computes their F1 accuracy. (2) ROUGE-L (R-L) which measures the lexical overlap between long-form answers from references and predictions. (3) Disambiguation-ROUGE (DR) score which is the geometric mean of Disambig-F1 and ROUGE-L and is used to assess the overall performance. Interested readers are requested to refer (Kim et. al., 2023) for a more detailed benchmarking study.

Table 5.2: Evaluation comparison of long-form QA on ambiguous questions on ASQA dataset (Kim et. al., 2023)

| Baseline | Model | D-F1 | R-L | DR |
|---|---|---|---|---|
| Fully Supervised | T5-Large Closed Book | 7.4 | 33.5 | 15.7 |
| Fully Supervised | T5-Large w/JPR | 26.4 | 43.0 | 33.7 |
| Fully Supervised | PaLM w/ Soft Prompt Tuning | 27.8 | 37.4 | 32.1 |
| Few Shot Prompting (5-shots) | PaLM | 25.3 | 34.5 | 29.6 |
| Few Shot Prompting | GPT-3 | 25.0 | 31.8 | 28.2 |

| | | | | |
|---|---|---|---|---|
| (5-shots) | | | | |
| Tree of Clarifications (ToC) | GPT-3 + RAC | 31.1 | 39.6 | 35.1 |
| Tree of Clarifications (ToC) | GPT-3 + RAC + TS | 32.4 | 40.0 | 36.0 |
| Tree of Clarifications (ToC) | GPT-3 + RAC + TS w/ Pruning | 33.7 | 39.7 | 36.6 |

## 5.3 Qualitative Evaluation

In this section, we look into the correctness of our framework, and evaluate if it is truly versatile enough to be considered as a strong contender for AI projects in the form of an industry standard methodology. The main purpose of this discussion is to ensure the data is correctly captured and transformed as it is meant to be, and the query resolution happens exactly as intended by the compute engine.

### 5.3.1 Question Answer Reasoning

Next, we evaluate how the Tree of Clarifications help in answering user queries on the basis of data stored. As an example, we will provide ambiguous questions that can have multiple answers based on their different possible interpretations. Clarify the given question into several disambiguated questions and provide short factoid answers to each question. Subsequently, summarize them into a detailed long-form answer of at least three sentences.

**Context**:

[1] <body> tag contains <nav> tag.

[2] <nav> tag contains 5 pairs of <ul> … </ul> tags.

[3] The text within 1st <ul> tag is “Home”

. . .

[7] The text within 5th <ul> tag is “Login/Logout”

**Question**: How to login into my user account?

**Disambiguations**:

DQ 1: What is the operation required?

DA 1: Login

DQ 2: Where is the login button in the given webpage?

DA 2: In the navbar

DQ 3: Where is the login button within the navbar?

DA 3: 5th <ul> tag within the navbar.

**Answer**: The user can login into their account by scrolling up on the navbar and clicking the fifth option in the list, namely "Login/Logout".

Here, as can be seen, Tree of Clarifications was able to resolve an ambiguous query by the user into multiple disambiguated question-answer pairs and return back the summarized long text answer by combining all the individual answers.

Interested readers are requested to refer to the original paper (Kim et. al., 2023) for a more detailed performance evaluation of ToC framework in Open Ended Question Answering.

### 5.3.2 SO(3) equivalence

Now, we are going to revisit one of the major research objective we defined for this work, namely-

"To formalize the notion of a neural network into an abstract structure while correctly capturing all the underlying properties."

Our desired objective here was to lift the input data into a standard modality that can be described mathematically in the form of an abstract structure. As per discussed methodology, we have been able to do so by utilizing VN-DGCNN (Deng et. al., 2021) that is SO(3)-equivariant which is the standard 3D rotation group under the operation of composition.

Some important properties of SO(3) are as follows:

- It is a non-abelian group.
- It is an invertible group, since every rotation has a unique inverse rotation.
- It is homeomorphic to the 3-dimensional real projective space $RP_3$.

Specifically, SO(3) is a subgroup of the general linear group consisting of all invertible linear transformations of the real 3-space $R^3$. Essentially, we have been able to model our complete machine learning pipeline in this form which was our intended objective.

From a performance perspective, as can be seen in table 5.5, VN-DGCNN clearly outperforms the other variants on point cloud classification tasks for point / mesh inputs on 3 train/test settings: z/z, z/SO(3) and SO(3)/SO(3), where z stands for data augmentation with rotations only around the z axis, and SO(3) for arbitrary rotations.

All rotations are generated on the fly at the training time, thereby comparing the equivariance-by-construction of vector neuron (VN) architecture with a learned-by-augmentation equivariance. Interested readers are requested to refer to the original paper (Deng et. al., 2021) for a more detailed performance analysis of vector neuron based point cloud processing backends for classification and other tasks.

Table 5.5 Classification accuracy on ModelNet40 dataset (VN-architecture) (Deng et. al., 2021)

| Method | z / z | z/SO(3) | SO(3)/SO(3) |
|---|---|---|---|
| PointNet | 85.9 | 19.6 | 74.7 |
| DGCNN | 90.3 | 33.8 | 88.6 |
| VN-PointNet | 77.5 | 77.5 | 77.2 |
| VN-DGCNN | 89.5 | 89.5 | 90.2 |

Here, it is noteworthy that the vector neuron network model has been tested on operations such as segmentation, point cloud reconstruction, etc. as well apart from the classification operation that we have employed in our framework. It remains an accessible option in use-cases where we might need those real-time operational capabilities in our data transformer rather than the classifier.

### 5.3.3 Data Structure Representation

Having discussed the file directory structure in our framework, we now re-look at our second objective for this project, and formally show how it serves the intended objective, namely:

“To correctly map the abstract structure in the form of a data structure for media content archival and editing.”

The output file received from the data transformer correctly stores the point cloud data in binary form alongwith the insights captured by the classification sequence passed into the language specification in the metadata dictionary. The language states traversed by the parser are recorded in the form of the metadata within the output file itself as discussed earlier in the methodology. Since we are using a formal language to understand and capture required insights from our common modality, our stated objective is met here as a direct consequence.

From a file representation internals perspective, this approach is flexible enough to allow the specific parameters such as what should be the exact metadata dictionary structure to be any data structure such as B-Tree, linked list, etc. It is completely up to the individual implementation requirements and is not hard-coded in any way.

### 5.3.4 Turing Completeness of the Language Specification

Now, we approach the final objective listed down for our project, namely,

“To formally evaluate if the aforementioned abstract structure is Turing recognizable.”

Here, if we look back at our language specification design, it is represented by a computable function which by construction is a formal language. We know as a matter of fact from the Church Turing thesis that every computable function is computable by a Turing machine and is subsequently a Turing recognizable language.

This holds significance in the context of our framework because it shows that we can, in essence, define the language specification to be any Turing recognizable language which can be something as simple as a regular grammar or all the way up to an unrestricted grammar as per the Chomsky hierarchy.

The Church-Turing thesis formally states that a function is considered as computable if and only if it can be solved by a Turing machine. This directly implies that any computable function would directly fit into our language specification for all practical purposes.

To conclude, our stated objective is met as a direct consequence of Church Turing thesis, since the stored metadata is directly obtained as a traversal path outcome from the defined language specification. Our language specification is always guaranteed to represent a computable function and hence the set of insights captured by it from the common modality are Turing recognizable as a direct result.

## 5.4 Summary

We have done a performance evaluation of our proposed framework by in-depth analysis of all the major building blocks and also compared how they perform with respect to the other available options. The main purpose of this evaluation was to ensure that our proposed framework as a whole is versatile and in-line with respect to current industrial trends. In the next chapter, we conclude this thesis work by looking back at our original research objectives and discussing what we have been able to achieve out of this work.

# CHAPTER 6: CONCLUSION & RECOMMENDATIONS

## 6.1 Discussion and Conclusion

As the famous Amara's law states: "We tend to overestimate the effect of a technology in the short run and underestimate the effect in the long run.", this study has been an attempt to showcase that while AI has a lot of potential, there is still work to be done in terms of how we adapt it in our projects in order to really add value and make the most of it. What we are essentially looking forward to in coming times is a paradigm shift in the technology sector at all layers- be it software application development, compiler design, hardware performance optimization or even in areas such as formal verification. We are now going to note down the conclusions that we have derived throughout this work below:

### 6.1.1 Formal Representation of Point Cloud Tokens

We have shown that VN-DGCNN is well capable of classifying while capturing all the point cloud data in a SO(3)-equivariant vector neuron modality. We have also discussed how the output file correctly captures the binary data as well as the associated insights in the form of metadata in a standard file format representation. This was essentially our 1st objective and as concluded in the previous chapter we have been able to build such an abstraction.

### 6.1.2 Data Structure representation for captured data

We have shown through our implementation details and supporting arguments in the previous chapter to conclude that the file format consisting of the binary point cloud data along with the captured metadata is rigid enough to capture all the insights captured from our abstract structure representation. This was our intended 2nd objective and thus, it has been achieved as well.

### 6.1.3 Expressive Power of the Formal Language Specification

We have also discussed how the proposed framework is capable of modeling any form of language representation by virtue of Church Turing thesis. As concluded with supporting arguments in the previous chapter, this is a direct implication of Church Turing thesis. This concludes our discussion around the stated 3rd objective as well.

Additionally, we have also completed a thorough literature survey around some major developments that have happened till recent times in the field of deep learning. We discussed deep feedforward networks, convolutional neural networks, RNNs, siamese networks and auto-encoders. We also covered deep generative models such as the Boltzmann machine, RBMs, VAEs, GANs, diffusion transformers, vision transformers and the more recent NeRFs. We then concluded our survey with a discussion on KANs. The motivation is that in one form or another, these are directly pluggable into our framework at multiple levels in the building blocks that we have discussed. This essentially completes our stated 4th objective as well.

## 6.2 Contribution

The proposed framework is capable of rapid prototyping and formal verification of software systems involving Machine Learning and AI. Formal verification methods have been well used in the hardware industry for chip verification since the manufacturing costs are very high for an incorrect or defective chip fabrication. However, to the best of our knowledge, this work is a first-of-its-kind attempt towards applying the same to the expanding field of Machine Learning and AI.

This framework is good enough to build easy to distribute starting templates for AI projects in the industry side of things, be it autonomous security systems, hospital reporting systems, etc. just to name a few.

The templates built with this framework are extensible and aimed towards making AI accessible without an expert involvement in day-to-day maintenance and operations.

This framework can be used as a medium to communicate with stakeholders in an AI project at all levels, and can be used as a general guideline or a design pattern for building AI solutions that are easy to deploy, use and further distribute.

Another note-worthy application for this framework is 3D reconstruction of monuments such as smart city planning and simulation, heritage sites preservation, gaming platforms, etc just to name a few.

This framework allows for AI development that is easy to explain and is free from any unwanted bias, since the implementer has the ability to hand-tune guardrails in the language specification, so that only those specific information that needs to be processed is indeed processed and personally identifiable information (PII) is strictly discarded or handled as per the standard regulations. This is specifically useful when building software applications for highly regulated industries such as healthcare and life sciences.

To conclude, the expressive power of this framework can be understood by the fact that we can virtually pick any ML/AI project in practice and model it in terms of this design. Whether we want to evaluate it, or showcase it to stakeholders in an easy to understand form, all of this can be done in one place itself.

## 6.3 Recommendations

### 6.3.1 Implementation Aspects

Further investigation needs to be done regarding how this framework will perform in a distributed computing scenario. Also, we need to investigate more specialized techniques specific to data types such as audio, video, text, etc. for potential optimizations. Further benchmarking needs to be done on how it performs under stress testing scenarios with more diverse datasets. Domain specific datasets such as healthcare, financial services, energy sector, education etc. need to be investigated further in order to improve the robustness and practical relevance of the framework.

### 6.3.2 Methodological Aspects

Better prompt resolving mechanisms need to be studied in the future. How software systems built using this framework can be made more predictable in terms of their explainability also needs to be investigated. Better evaluation benchmarks for this framework specific to more specialized use cases and data types also needs to be explored in the future. Exploring representations other than point cloud representations that are relatively easier to implement is also a potential area of study. Improvements involving algorithms around language specification parsing and related optimizations in this regard needs to be explored as well. Better performant classification techniques for point cloud data remains a potential area of study as well.

# REFERENCES


[Emergency sign] [online image] (n.d.) Available at: https://help.salesforce.com/s/articleView?id=release-notes.rn_einstein_vision_ocr_ga.htm&release=226&type=5 [Accessed: 23rd October 2024]

Atzmon, M., Maron, H., Lipman, Y., (2018) Point Convolutional Neural Networks by Extension Operators. ACM Trans. Graph. 37, 4, Article 71 (July 2018), 12 pages. [online] Available at: https://doi.org/10.1145/3197517.3201301 [Accessed 13th October 2024]

Bank, D., Koenigstein, N., Giryes, R., (2023) Autoencoders. In Rokach, L., Maimon, O., Shmueli, E., (eds.). Machine learning for data science handbook. pp. 353–374.

Bhat, S.F., Birkl, R., Wofk, D., Wonka, P., Müller, M., (2023) Zoedepth: Zero-shot transfer by combining relative and metric depth. [online] Available at: https://arxiv.org/abs/2302.12288 [Accessed 26th September 2024]

Brock, A., Lim, T., Ritchie, J.M., Weston, N.J., (2016) Generative and Discriminative Voxel Modeling with Convolutional Neural Networks. [online] Available at: https://arxiv.org/abs/1608.04236 [Accessed 6th November 2024]

Caulfield, B., (2025) CES 2025: AI Advancing at 'Incredible Pace,' NVIDIA CEO Says. [online] Available at: https://blogs.nvidia.com/blog/ces-2025-jensen-huang [Accessed 15th January 2025]

Chang, Z., Koulieris, G.A., Shum, H.P.H., (2023) On the Design Fundamentals of Diffusion Models: A Survey. [online] Available at: https://arxiv.org/abs/2306.04542 [Accessed 21st September 2024]

Cheng , A., Yin, H., Fu, Y., Guo, Q., Yang, R., Kautz, J., Wang, X., Liu, S., (2024) SpatialRGPT: Grounded Spatial Reasoning in Vision Language Models. [online] Available at: https://arxiv.org/abs/2406.01584 [Accessed 3rd November 2024]

Chicco, D., (2020) Siamese neural networks: an overview. Artificial Neural Networks, Methods in Molecular Biology, vol. 2190 (3rd ed.), New York City, New York, USA: Springer Protocols, Humana Press, pp. 73–94, doi:10.1007/978-1-0716-0826-5_3, ISBN 978-1-0716-0826-5, PMID 32804361, S2CID 221144012

Choy, C., Dong, W., Koltun, V., (2020) Deep Global Registration. [online] Available at: https://arxiv.org/abs/2004.11540 [Accessed 23rd September 2024]

Degroot, M., Schervish, M., (2014) Probability and Statistics, 4th Edition, Pearson.

Deng, C., Litany, O., Duan, Y., Poulenard, A., Tagliasacchi, A., Guibas, L., (2021) Vector Neurons: A General Framework for SO(3)-Equivariant Networks. [online] Available at: https://arxiv.org/abs/2104.12229 [Accessed 2nd October 2024]

Deng, H., Birdal, T., llic, S., (2018) PPFNet: Global Context Aware Local Features for Robust 3D Point Matching. [online] Available at: https://arxiv.org/abs/1802.02669 [Accessed 3rd October 2024]

Dosovitskiy , A., Beyer , L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N., (2021) An Image is worth 16x16 Words: Transformers for Image Recognition at Scale. [online] Available at: https://arxiv.org/abs/2010.11929 [Accessed 4th October 2024]

Dummit, D.S., Foote, R.M., (2004) Abstract Algebra, 3rd Edition, John Wiley & Sons Inc.

Dwivedi, S.K., Sun, Y., Patel, P., Feng, Y., Black, M.J., (2024) TokenHMR: Advancing Human Mesh Recovery with a Tokenized Pose Representation. [online] Available at: https://arxiv.org/abs/2404.16752 [Accessed 28th September 2024]

Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., Larson, J., (2024) From Local to Global: A Graph RAG Approach to Query-Focused Summarization. [online] Available at: https://arxiv.org/abs/2404.16130 [Accessed 1st October 2024]

Esser, P., Kulal, S., Blattmann, A., Entezari, R., Muller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., Podell, D., Dockhorn, T., English, Z., Lacey, K., Goodwin, A., Marek, Y., Rombach, R., (2024) Scaling Rectified Flow Transformers for High-Resolution Image Synthesis. [online] Available at: https://arxiv.org/abs/2403.03206 [Accessed 4th October 2024]

Fischer, A., Igel, C., (2012) An Introduction to Restricted Boltzmann Machines, Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, Lecture Notes in Computer Science, vol. 7441, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 14–36, doi:10.1007/978-3-642-33275-3_2, ISBN 978-3-642-33274-6

Gao, R., Holynski, A., Henzler, P., Brussee, A., Martin-Brualla, R., Srinivasan, P., Barron, J.T., Poole, B., (2024) CAT3D: Create Anything in 3D with Multi-View Diffusion Models. [online] Available at: https://arxiv.org/abs/2405.10314 [Accessed 23rd September 2024]

Geiger, A., Lenz, P., Stiller, C., Urtasun, R., (2013) Vision meets robotics: The kitti dataset. [online] Available at: https://www.cvlibs.net/publications/Geiger2013IJRR.pdf [Accessed 6th November 2024]

Gezawa, A.S., Zhang, Y., Wang, Q., Yunqi, L., (2020) A review on deep learning approaches for 3d data representations in retrieval and classifications. [online] Available at: https://ieeexplore.ieee.org/document/9043500 [Accessed 30th October 2024]

Goodfellow, I., Bengio, Y., Courville, A., (2016) Deep Learning, The MIT Press.

Gu, J., Zhao, N., Xiong, W., Liu, Q., Zhang, Z., Zhang, H., Zhang, J., Jung, H., Wang, Y., Wang, X.E., (2024) SwapAnything: Enabling Arbitrary Object Swapping in Personalized Visual Editing. [online] Available at: https://arxiv.org/abs/2404.05717 [Accessed 10th September 2024]

Hatamizadeh , A., Song, J., Liu, G., Kautz, J., Vahdat, A., (2024) DiffiT: Diffusion Vision Transformers for Image Generation. [online] Available at: https://arxiv.org/abs/2312.02139 [Accessed 3rd November 2024]

Hengshuang, Z., Jiang, L., Jiaya, J., Torr, P., Koltun, V., (2020) Point Transformer. [online] Available at: https://arxiv.org/abs/2012.09164 [Accessed 10th October 2024]

Hinton, G., (2009) Deep belief networks. Scholarpedia. 4 (5): 5947. Bibcode:2009SchpJ...4.5947H. doi:10.4249/scholarpedia.5947

Kim, G., Kim, S., Jeon, B., Park, J., Kang, J. (2023) Tree of Clarifications: Answering Ambiguous Questions with Retrieval-Augmented Large Language Models. [online] Available at: https://arxiv.org/abs/2310.14696 [Accessed 2nd September 2024]

Kingma, D.P., Welling, M., (2022) Auto-Encoding Variational Bayes. [online[ Available at: https://arxiv.org/abs/1312.6114 [Accessed 27th September 2024]

Klapheke, D., (2023) What is Frame Rate in Video? [online] Available at: https://www.switcherstudio.com/blog/what-is-frame-rate [Accessed 30th October 2024]

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B., (2018a) PointCNN: Convolution On X-Transformed Points. [online] Available at: http://papers.nips.cc/paper/7362-pointcnn-convolution-on-x-transformed-points.pdf [Accessed 5th October 2024]

Liu, A., Lin, C., Liu, Y., Long, X., Dou, Z., Guo, H., Luo, P., Wang, W., (2024) Part123: Part-aware 3D Reconstruction from a Single-view Image. [online] Available at: https://arxiv.org/abs/2405.16888 [Accessed 3rd October 2024]

Liu, J., Luo, J., Shah, M., Recognizing Realistic Actions from Videos in the Wild. IEEE International Conference on Computer Vision and Pattern Recognition(CVPR), 2009. [online] Available at: https://www.crcv.ucf.edu/papers/cvpr2009_liu1.pdf [Accessed 20th September 2024]

Liu, S., Zhang, M., Kadam, P., Kuo, C.C.J., (2021) 3D Point Cloud Analysis: Traditional, Deep Learning and Explainable Machine Learning Methods, Springer.

Liu, Y., Zhang, K., Li, Y., Yan, Z., Gao, C., Chen, R., Yuan, Z., Huang, Y., Sun, H., Gao, J., He, L., Sun, L., (2024) Sora: A Review on Background, Technology, Limitations, and Opportunities of Large Vision Models. [online] Available at: https://arxiv.org/abs/2402.17177 [Accessed 29th October 2024]

Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M., Hou, T.Y., Tegmark, M., (2024) KAN: Kolmogorov-Arnold Networks. [online] Available at: https://arxiv.org/abs/2404.19756 [Accessed 10th October 2024]

Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng R., (2020) NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. [online] Available at: https://arxiv.org/abs/2003.08934 [Accessed 16th September 2024]

Mou, C., Cao, M., Wang, X., Zhang, Z., Shan, Y., Zhang, J., (2024) ReVideo: Remake a Video with Motion and Content Control. [online] Available at: https://arxiv.org/abs/2405.13865 [Accessed 28th September 2024]

N.C., Bruce, (2015) Advanced Linear Algebra, 2nd Edition, CRC Press.

Potje, G., Cadar, F., Araujo, A., Martins, R., Nascimento, E.R. (2024) XFeat: Accelerated Features for Lightweight Image Matching. [online] Available at: https://arxiv.org/abs/2404.19174 [Accessed 10th October 2024]

Qi, C.R., Su, H., Mo, K., Guibas, L.J., (2017b) PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. [online] Available at: https://arxiv.org/abs/1612.00593 [Accessed 15th September 2024]

Qi, C.R., Yi, L., Su, H., Guibas, L.J.. (2017c) PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. [online] Available at: https://arxiv.org/abs/1706.02413 [Accessed 14th September 2024]

Raut, G., Patole, A., (2023) End-to-End 3D Object Detection using LiDAR Point Cloud. [online] Available at: https://arxiv.org/abs/2312.15377 [Accessed 15th September 2024]

Ravi, N., Gabeur, V., Hu, Y.T., Hu, R., Ryali, C., Ma, T., Khedr, H., Rädle, R., Rolland, C., Gustafson, L., Mintun, E., Pan, J., Alwala, K.V., Carion, N., Wu, C.Y., Girshick, R., Dollár, P., Feichtenhofer, C., (2024) SAM 2: Segment Anything in Images and Videos. [online] Available at: https://arxiv.org/abs/2408.00714 [Accessed 15th November 2024]

Rexer, K., Siegel, E., (2023) 2023 Data Science Survey Preliminary Results. [online] Available at: https://drive.google.com/file/d/1Mz3WmtcvUl-00gaT2XKCxdE5-pqbOOjz/view [Accessed 20th September 2024]

Rich, E., (2008) Automata, Computability and Complexity: Theory and Applications, Pearson.

Royden, H., Fitzpatrick, P., (2023) Real Analysis and Measure Theory, 5th Edition, Pearson.

Ryseff, J., Bruhl, B.F.D., Newberry, S.J., (2023) The Root Causes of Failure for Artificial Intelligence Projects and How They Can Succeed. [online] Available at: https://www.rand.org/pubs/research_reports/RRA2680-1.html [Accessed 27th September 2024]

Shao, S., Pei, Z., Chen, W., Wu, X., Li, Z., (2023a) Nddepth: Normal-distance assisted monocular depth estimation. [online] Available at: https://arxiv.org/abs/2311.07166 [Accessed 29th September 2024]

Shao, S., Pei, Z., Wu, X., Liu, Z., Chen, W., Li, Z., (2023b) Iebins: Iterative elastic bins for monocular depth estimation. [online] Available at: https://arxiv.org/abs/2309.14137 [Accessed 1st October 2024]

Sherrington, D., Kirkpatrick, S., (1975) Solvable Model of a Spin-Glass, Physical Review Letters, 35 (35): 1792–1796, doi:10.1103/PhysRevLett.35.1792

Silberman, N., Hoiem, D., Kohli, P., Fergus, R., Indoor segmentation and support inference from rgbd images. In ECCV, 2012.

Simonyan, K., Zisserman, A., (2014) Two-Stream Convolutional Networks for Action Recognition in Videos. [online] Available at: https://arxiv.org/abs/1406.2199 [Accessed 27th September 2024]

Stamford, C., (2018) Gartner Says Nearly Half of CIOs Are Planning to Deploy Artificial Intelligence. [online] Available at: https://www.gartner.com/en/newsroom/press-releases/2018-02-13-gartner-says-nearly-half-of-cios-are-planning-to-deploy-artificial-intelligence [Accessed 3rd October 2024]

Stelmakh, I., Luan, Y., Dhingra, B., Chang, M., (2022) ASQA: Factoid Questions meet Long-Form Answers. [online] Available at: https://arxiv.org/abs/2204.06092 {Accessed 2nd September 2024]

Tian, Y., Zhao, J., Dong, H., Xiong, J., Xia, S., Zhou, M., Lin, Y., Cambronero, J., He, Y., Han, S., Zhang, D., (2024) SpreadsheetLLM: Encoding Spreadsheets for Large Language Models. [online] Available at: https://arxiv.org/abs/2407.09025v1 [Accessed 21st September 2024]

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., (2017) Attention is All you Need. [online] Available at: https://arxiv.org/abs/1706.03762 [Accessed 2nd September 2024]

Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., Ding, G., (2024) YOLOv10: Real-Time End-to-End Object Detection. [online] Available at: https://arxiv.org/abs/2405.14458 [Accessed 3rd October 2024]

Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M., (2019) Dynamic Graph CNN for Learning on Point Clouds. [online] Available at: https://arxiv.org/abs/1801.07829 [Accessed 3rd October 2024]

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J., (2015) 3D Shapenets: A Deep Representation for Volumetric Shapes. [online] Available at: https://arxiv.org/abs/1406.5670 [Accessed 3rd September 2024]

Xiao, B., Wu, H., Xu, W., Dai, X., Hu, H., Lu, Y., Zeng, M., Liu, C., Yuan, L., (2023) Florence-2: Advancing a Unified Representation for a Variety of Vision Tasks. [online] Available at: https://arxiv.org/abs/2311.06242 [Accessed 25th September 2024]

Yang, L., Kang, B., Huang, Z., Zhao, Z., Xu, X., Feng, J., Zhao, H., (2024) Depth Anything V2. [online] Available at: https://arxiv.org/abs/2406.09414 [Accessed 15th October 2024]

Zhao, W., Rao, Y., Liu, Z., Liu, B., Zhou, J., Lu, J., (2023) Unleashing text-to-image diffusion models for visual perception. [online] Available at: https://arxiv.org/abs/2303.02153 [Accessed 6th October 2024]

Zhou, L., Tang, T., Hao, P., He, Z., Ho, K., Gu, S., Hou, W., Hao, Z., Sun, H., Zhan, K., Jia, P., Lang, X., Liang, X., (2024) UA-Track: Uncertainty-Aware End-to-End 3D Multi-Object Tracking. [online] Available at: https://arxiv.org/abs/2406.02147 [Accessed 16th October 2024]

Zhu, L., Li, Y., Liu, N., Peng, H., Yang, D., Shlizerman, I.K., (2024) M&M VTO: Multi-Garment Virtual Try-On and Editing. [online] Available at: https://arxiv.org/abs/2406.04542 [Accessed 13th October 2024]

# APPENDIX A: PROJECT CODE

## Frontend Component Code

1. TobyFileUploadAndValidator.html

```
<template>

   <lightning-card title="Text Grammar Validator"
icon-name="action:approval">

       <div style="max-width:30%;" class="slds-m-around_medium">

           <lightning-file-upload

               label="Upload a Text File"

               accept=".html, .json"

               multiple="false"

               onuploadfinished={handleFileUpload}>

           </lightning-file-upload>

       </div><hr style="margin:0px!important;"/>

                   <lightning-layout if:true={compliantRulesExists}
multiple-rows="true">

               <lightning-layout-item padding="around-small"
size="12">

                   <lightning-layout>

                       <lightning-layout-item padding="around-small"
size="3">

                           <div class="page-section page-right">

                               <template if:true={fileContent}>

                                   <div
class="slds-m-around_medium">

                                       <strong><h3>File Content
Preview</h3></strong><br/>
```

```
                                        <div class="file-preview">
                                            <div style="white-space:
pre-line">
                                                {fileContent}
                                            </div>
                                        </div>
                                    </div>
                                </template>
                            </div>
                        </lightning-layout-item>
                        <lightning-layout-item padding="around-small"
size="6">
                            <div class="page-section page-main">
                                <template
if:true={compliantRulesExists}>
                                    <div
class="slds-m-around_medium">
                                        <strong><h3>Compliant Grammar
Rules:</h3></strong><br/>
                                        <ul>
                                            <template
for:each={compliantRules} for:item="rule">
                                                <strong
key={rule.Id}>{rule.Name}:</strong> <lightning-formatted-rich-text
key={rule.Id}
value={rule.Description__c}></lightning-formatted-rich-text>
                                                <em
key={rule.Id}>Matched {rule.matchCount} time(s).</em>
                                                <br
key={rule.Id}/><br key={rule.Id}/>
```

```
                                        </template>
                                    </ul>
                                </div>
                            </template>
                        </div>
                    </lightning-layout-item>
                    <lightning-layout-item padding="around-small"
size="3">
                        <div class="page-section page-right">
                            <lightning-select
                                name="regex"
                                label="Rule to replace"
                                value={value}
                                options={options}
                                onchange={handleChange}
                            ></lightning-select>
                            <lightning-input
                                type="text"
                                label="Replacement Text"
                                placeholder="Enter replacement
text"

onchange={handleReplacementChange}
                            ></lightning-input>
                            <div class="slds-grid
slds-grid_align-center slds-p-top_small">
                                <div class="slds-col">
```

```
                                        <lightning-button
variant="brand-outline" label="Replace Text"
                                        title="Replace Text"
onclick={handleClick}
                                        ></lightning-button>
                                    </div>
                                </div>
                            </div>
                        </lightning-layout-item>
                    </lightning-layout>
                </lightning-layout-item>
            </lightning-layout>
    </lightning-card>
</template>
```

2.TobyFileUploadAndValidator.js

```
/*
************************************************************
File Name    : TobyFileUploadAndValidator.js
Created Date       : Jan 12, 2025
@description       : This JS class is used for File metadata Upload,
Validation and Replace Operations.
@author            : Sanket Anand
Modification Log:
Ver   Date         Author                              Modification
1.0   12-01-2025   Sanket Anand                    Initial Version
************************************************************
*/
```

```
import { LightningElement, track } from 'lwc';

import fetchGrammarRules from
'@salesforce/apex/HtmlValidatorController.fetchGrammarRules';

import getTextFileContent from
'@salesforce/apex/HtmlValidatorController.getTextFileContent';

import createFileLink from
'@salesforce/apex/HtmlValidatorController.createFileLink';


export default class TobyFileUploadAndValidator extends
LightningElement {

   @track validationResult = ''; // Validation rule triggered (If
any)

   @track compliantRules = []; //Set of rules matched in the file

   @track fileContent = ''; // Binary content of the file uploaded

   @track modifiedContent = ""; // Stores the modified file content

   replacementText = ""; // User-provided replacement text

   rules = []; // Set of all rules to match with

   value = ''; // Replacement text provided by the user

   contentDocumentID; //ID of File uploaded

   options = [{ label: 'Choose One...', value: '' }]; //Values for
the Dropdown for rule to replace


   /*

   ************************************************************

   @Method Name    : compliantRulesExists

   @author         : Sanket Anand

   @description    : Getter function to check if compliant rules
exist or not.
```

```
    ************************************************************
    */
    get compliantRulesExists(){
        return ((this.compliantRules.length > 0) ||
(this.fileContent.length > 0));
    }

    /*
    ************************************************************
    @Method Name    : handleChange
    @author         : Sanket Anand
    @description    : Method to select rule to replace in text
    ************************************************************
    */
    handleChange(event) {
        this.value = event.detail.value;
    }

    /*
    ************************************************************
    @Method Name    : connectedCallback
    @author         : Sanket Anand
    @description    : Method to fetch all the DFA rules to match file
against
    ************************************************************
    */
    connectedCallback() {
```

```
        // Fetch grammar rules on component load
        fetchGrammarRules()
            .then((result) => {
                this.rules = JSON.parse(result);
            })
            .catch((error) => {
                console.error('Error fetching grammar rules:',
error);
            });
    }

    /*
    ********************************************************
    @Method Name    : handleClick
    @author         : Sanket Anand
    @description    : Method to validated uploaded file against the
rules
    ********************************************************
    */
    handleClick(){
        const regex = new RegExp(this.value, "g"); // Create regex
with global flag
        this.fileContent = this.fileContent.replace(regex,
this.replacementText); // Replace text
        this.validateHtml(this.fileContent);
    }
```

```
    /*
    ********************************************************
    @Method Name    : handleFileUpload
    @author         : Sanket Anand
    @description    : Method to fetch and validate binary content of
the file content
    ********************************************************
    */
    handleFileUpload(event) {
        this.options = [{ label: 'Choose One...', value: '' }];
        const uploadedFiles = event.detail.files;
        if (uploadedFiles && uploadedFiles.length > 0) {
            const file = uploadedFiles[0];
            this.contentDocumentID = file.documentId;
            getTextFileContent({contentVersionId:
file.contentVersionId}).then((resultApex) => {
                this.fileContent = resultApex; // Store the file
content for preview
                this.validateHtml(resultApex);
            });
        } else {
            this.validationResult = 'No file was uploaded.';
        }
    }


    /*
    ********************************************************
```

```
    @Method Name    : handleReplacementChange
    @author         : Sanket Anand
    @description    : Method to update text with the replacement text
entered
    ********************************************************
    */
    handleReplacementChange(event) {
        this.replacementText = event.target.value;
    }

    /*
    ********************************************************
    @Method Name    : validateHtml
    @author         : Sanket Anand
    @description    : Method to validate HTML with standard DOMParser
API
    ********************************************************
    */
    validateHtml(content) {
        try {
            const parser = new DOMParser();
            const doc = parser.parseFromString(content, 'text/html');
            const errors = doc.querySelectorAll('parsererror');

            if (errors.length > 0) {
                this.validationResult = `Invalid HTML. Found
${errors.length} error(s).`;
```

```
            } else {
                this.validationResult = 'The HTML is valid.';
                this.checkGrammarCompliance(content);
            }
        } catch (error) {
            this.validationResult = `Error parsing HTML:
${error.message}`;
        }
    }

    /*
    ************************************************************
    @Method Name    : checkGrammarCompliance
    @author         : Sanket Anand
    @description    : Method to check for compliant grammar rules and
call backend function for creating linkeages
    ************************************************************
    */
    checkGrammarCompliance(htmlContent) {
        const compliantRules = this.rules.map((rule) => {
            const regex = new RegExp(rule.Regex_Expression__c, 'g');
            const matches = htmlContent.match(regex) || [];
            return {
                ...rule,
                matchCount: matches.length, // Count the matches
            };
```

```
            }).filter((rule) => rule.matchCount > 0); // Only include
        rules with matches
            const applicableRule = compliantRules.map((rule) => {
                this.options.push({label: rule.Name, value:
        rule.Regex_Expression__c});
                return rule.Id;
            });
            this.compliantRules = compliantRules;
            if (compliantRules.length === 0) {
                this.validationResult += ' No grammar rules matched.';
            } else {
                this.validationResult += ` Complies with
        ${compliantRules.length} grammar rule(s).`;
                createFileLink({fileId: this.contentDocumentID,
        rulesMatched: applicableRule, finalResult:
        JSON.stringify(this.compliantRules)}).then(() => {
                    console.log('records saved.')
                });
            }
        }
    }
```

3. TobyFileUploadAndValidator.css

```
.custom-container{
    color : black;
}
.toby_container{
    min-height: 600px;
}
```

```
.c-container {
    border: 1px solid #d8dde6;
    margin: 10px 0 20px;
}
.page-section {
    border: solid 1px #ccc;
    padding: 1rem;
}
.page-header,
.page-footer {
    height: 50px;
}
.page-main {
    background: #f8f8f8;
}
.page-left,
.page-right {
    background: #f0efef;
}
```

4. TobyFileUploadAndValidator.js-meta.xml

```
<?xml version="1.0"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>62.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
```

```
        <target>lightning__AppPage</target>
        <target>lightning__HomePage</target>
        <target>lightning__RecordPage</target>
        <target>lightning__Tab</target>
    </targets>
</LightningComponentBundle>
```

## Backend Code

1. HtmlValidatorController.cls

```
/*
************************************************************
Apex Class Name    : HtmlValidatorController
Created Date       : Jan 12, 2025
@description       : This class is used for File metadata Upload and
Validation Operations.
@author            : Sanket Anand
Modification Log:
Ver   Date         Author                                Modification
1.0   12-01-2025   Sanket Anand                       Initial Version
************************************************************
*/
public without sharing class HtmlValidatorController {

    /*
    ************************************************************
    @Method Name    : fetchGrammarRules
    @author         : Sanket Anand
```

```
    @description    : Method to fetch all the DFA rules to match file
against
    @param          : void
    @return         : List of Rules in JSON Serialized form
    ********************************************************
    */
    @AuraEnabled(cacheable=true)
    public static String fetchGrammarRules() {
        List<DFA_Rule__c> rules = [
            SELECT Id, Regex_Expression__c, Description__c, Name
            FROM DFA_Rule__c
        ];
        return JSON.serialize(rules);
    }

    /*
    ********************************************************
    @Method Name    : getTextFileContent
    @author         : Sanket Anand
    @description    : Method to get contents of the file uploaded
    @param          : contentVersionId: Id of file
    @return         : Binary content of file
    ********************************************************
    */
    @AuraEnabled(cacheable=true)
    public static String getTextFileContent(String contentVersionId) {
```

```
        //Check if file ID is not blank
        if (String.isEmpty(contentVersionId)) {
            throw new IllegalArgumentException('Content Version ID cannot
be null or empty.');
        }

        // Query the Content Version associated with the Content Document
ID
        ContentVersion contentVersion = [
            SELECT Id, Title, FileType, VersionData
            FROM ContentVersion
            WHERE Id =: contentVersionId
        ];

        // Retrieve and return the file content as a String
        Blob fileBlob = contentVersion.VersionData;
        return fileBlob.toString();
    }

    /*
    ********************************************************
    @Method Name    : createFileLink
    @author         : Sanket Anand
    @description    : Method to create user
    @param          : fileId: ID of file, rulesMatched: IDs of rules
matched, finalResult: JSON with matched rules details
```

```
    @return          : void
    ********************************************************
    */
    @AuraEnabled(cacheable=false)
    public static void createFileLink(String fileId, String[] rulesMatched,
String finalResult) {

        //Check if File ID is present
        if (String.isEmpty(fileId)) {
            throw new IllegalArgumentException('File ID cannot be null or
empty.');
        }

        //Create Parent File Tag for matched rules
        List<HtmlTagRecord__c> htcs = new List<HtmlTagRecord__c>();
        HtmlTagRecord__c ht = new HtmlTagRecord__c();
        ht.Tags__c = finalResult;
        insert ht;

        //Link file Tags with file.
        List<ContentDocumentLink> cdl = new List<ContentDocumentLink>();
        ContentDocumentLink cd0 = new ContentDocumentLink();
        cd0.LinkedEntityId = ht.Id;
        cd0.ContentDocumentId = fileId;
        cdl.add(cd0);

        //Create Child File Tags for matched rules
```

```
        for(String s: rulesMatched){
            HtmlTagRecord__c htc = new HtmlTagRecord__c();
            htc.Tags__c = finalResult;
            htc.DFA_Rule__c = s;
            htc.Parent_Record__c = ht.Id;
            htcs.add(htc);
            ContentDocumentLink cd = new ContentDocumentLink();
            cd.LinkedEntityId = s;
            cd.ContentDocumentId = fileId;
            cdl.add(cd);
        }
        //Insert child file tags
        if(htcs.size() > 0){
            insert htcs;
        }
        //Insert child file tag links to the file
        if(cdl.size() > 0){
            insert cdl;
        }
    }
}
```

**APPENDIX B: RESEARCH PROPOSAL**

# Digital Footprinting, Analysis and Archival of Digital Media Content using Vision Transformers and Related Techniques

Sanket Anand

Research Proposal for
Master of Science in Computer Science

Liverpool John Moores University & upGrad

March 2024

# Abstract

Deep Learning and Generative AI in particular has seen remarkable advancements in recent years with the advent of Diffusion Transformers. The advancements made in this domain finds applications in far fetched areas such as Multimodal AI, Visual Question Answering, Explainable AI, Autonomous Systems, etc just to name a few. While the current literature is more inclined towards formalising the whole concept of a Neural Network as a system of Bayesian Probabilities whose change is guided by Derivatives or fundamentals of Linear Algebra in general, this thesis work attempts to formalise the general concept of a Neural Network into a Theoretical Framework the core of which will be an Abstract Structure. Such a construction could potentially open up the possibility of exploring a Machine Learning Model as a Software System that can be thoroughly verified using Formal Methods. It will also open up the possibility wherein a Machine Learning Model can be studied and analysed using algebraic, topological and analytical methods. Finally, we then conclude the thesis work by demonstrating how this proposed Abstract Structure fits in the form of a Composite Data Structure that can be used for Data Archival, Footprinting and Analysis in Machine Learning Applications.

## 1. Background

In recent years, Deep Learning and Generative AI has seen major breakthroughs, some of them being Diffusion Models (Chang, Ziyi; Koulieris, George Alex; Shum, Hubert P. H. et al., 2023) and Transformers (Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N; Kaiser, Łukasz; Polosukhin, Illia et al., 2017). Today, it finds applications in diverse areas such as Natural Language Processing, Computer Vision, Multimodal AI, etc. just to name a few.

Being able to constantly fine tune the output of such generative models has been an active area of study and much progress has been made around the same. Today, they are capable of accurately generating output that is indistinguishable from human generated content.

Generative Models, by nature are Resource Hungry and a lot of recent research has been focused on enabling them for edge devices. There has also been a good amount of effort in development of completely new Deep Learning Architectures fundamentally inspired from the same Transformer Based Architecture.

There has been a considerable amount of effort in advancing RAG Systems that improvise a Trained Model with Domain Specific or Real Time Ground Truth, thereby preventing Hallucinations (Gangwoo Kim, Sungdong Kim, Byeongguk Jeon, Joonsuk Park, Jaewoo Kang et al., 2023).

In the domain of Media Analytics, there has been substantial progress around 3D Reconstruction of Images and Media Content using Generative AI Techniques. Good amount of research has been done around related downstream applications such as Depth Estimation (Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, Hengshuang Zhao et al., 2024), Temporal Spatial Reasoning (An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, Sifei Liu et al., 2024), Pose Estimation (Sai Kumar Dwivedi, Yu Sun, Priyanka Patel, Yao Feng, Michael J. Black et al., 2024), Virtual Try On (Luyang Zhu, Yingwei Li, Nan Liu, Hao Peng, Dawei Yang, Ira Kemelmacher-Shlizerman et al., 2024), etc just to name a few.

There has also been a considerable amount of work in related areas such as Explainable AI, Graph Based RAG Systems (Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Jonathan Larson et al., 2024), Spreadsheet LLM (Yuzhang Tian, Jianbo Zhao, Haoyu Dong, Junyu Xiong, Shiyu Xia, Mengyu Zhou, Yun Lin, José Cambronero, Yeye He, Shi Han, Dongmei Zhang et al., 2024), etc.

## 2. Related Work

Deep Learning and Generative AI has seen tremendous progress in the last decade. There have been multiple notable achievements such as Transformers (Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N; Kaiser, Łukasz; Polosukhin, Illia et al., 2017), Boltzmann Machines (Sherrington, David; Kirkpatrick, Scott et. al, 1975), Restricted Boltzmann Machines (Fischer, Asja; Igel, Christian et al., 2012), Auto Encoders (Bank, Dor; Koenigstein, Noam; Giryes, Raja et al., 2023), Variational AutoEncoders (Kingma, Diederik P.; Welling, Max et al., 2022), Deep Belief Network (Hinton G et al., 2009) and the recently introduced Kolmogorov-Arnold Network (Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, Max Tegmark et al., 2024) just to name a few.

The field of Media Analytics and Generative AI in itself has been widely impacted by the emergence of Diffusion Models. There has been an emergence of multiple models powered by the same in the recent years such as SORA (Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, Lifang He, Lichao Sun et al., 2024), Florence (Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, Lu Yuan et al., 2023), etc just to name a few. Region Based Media Editing has seen advancements using Token Based Approaches (Sai Kumar Dwivedi, Yu Sun, Priyanka Patel, Yao Feng, Michael J. Black et al., 2024).

There has been multiple works around Gaussian Splatting that finds applications in use cases such as 3D Reconstruction (Ruiqi Gao, Aleksander Holynski, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul Srinivasan, Jonathan T. Barron, Ben Poole et al., 2024), Virtual

Try On (Luyang Zhu, Yingwei Li, Nan Liu, Hao Peng, Dawei Yang, Ira Kemelmacher-Shlizerman et al., 2024), Pose Estimation (Sai Kumar Dwivedi, Yu Sun, Priyanka Patel, Yao Feng, Michael J. Black et al., 2024), etc.

Neural Radiance Fields (NeRF) is another technique that has found profound applications in recent literature around 3D Reconstruction in general. Some other Topics under active research in recent years include Depth Estimation (Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, Hengshuang Zhao et al., 2024), Instance and Region Based Segmentation (Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, Guiguang Ding et al., 2024) as well as RaDAR, LiDAR Point Clouds (Gaurav Raut, Advait Patole et al., 2023).

There has also been a good amount of work in Fine Tuning such models with real time domain information with the help of RAG Systems, in inorder to answer or operate upon ambiguous prompts (Gangwoo Kim, Sungdong Kim, Byeongguk Jeon, Joonsuk Park, Jaewoo Kang et al., 2023).

## 3. Research Questions

This thesis tries to answer the following questions:

1. Can we accurately model the stream of tokens from our Neural Network into an Abstract Structure?
2. What is the most accurate Abstract Structure that can correctly capture the token value as well as its underlying relationship with other tokens.
3. Does this Abstract Structure respect results from Church Turing Thesis in the sense it is equivalent to a Turing Recognizable Language?
4. Can we subsequently implement this Abstract Structure in the form of a Data Structure for Media Analytics Applications?

## 4. Aim and Objectives

This work tries to formalise the notion of a Neural Network into an Abstract Structure that can be further extended in the form of a Data Structure for archival and editing Media Content.

Objectives:

- To Formalise the notion of a Neural Network into an Abstract Structure while correctly capturing all the underlying properties.
- To correctly map the Abstract Structure in the form of a Data Structure for Media Content Archival and Editing.
- To formally evaluate if the aforementioned Abstract Structure is Turing Recognizable.

## 5. Significance of the Study

Deep Learning and Generative AI in particular is a field under active research. While the current methods rely on considering a Neural Network as a system of Bayesian Probabilities, there has been little to no literature work that has happened around exploring the same from an analytical perspective.

This work tries to fill in this gap by introducing methods and techniques exploring the same from a pure mathematics lens. This study could serve as a fundamental tool around classic geometric, algebraic and topological properties of a Neural Network.

In terms of applications, we then show how the proposed structure can also be implemented as an efficient data structure for archival and editing of media content in AI use cases.

## 6. Scope of the Study

The scope of this thesis work is defined as follows:

- The thesis work has to be completed within 17 weeks post research proposal submission.
- The experimentation work related to the proposed research will be conducted using open-source compiler tools.
- The proposed research will be conducted on a local GPU, namely RTX 3060.

## 7. Research Methodology

This work focuses on the formalisation of the concept of a Neural Network into an Abstract Structure which preserves all the properties in the form of a Token, and also preserves all the relationships between those tokens. It is then shown how this formalisation can also be implemented in the form of a data structure for efficient archival and editing of Media data in AI Applications.

### 7.1 Dataset Description

This work makes use of two datasets-

- **UCF11**: Introduced by (Jingen Liu, Jiebo Luo and Mubarak Shah, et al., 2009), this dataset contains 11 action categories with 25 groups under each category containing at least 4 video clips for each group. The dataset is widely used in human action recognition tasks. The dataset is specifically known for its complexity in terms of camera stability, scaling, object and pose appearance, background and illumination variations, etc.
- **ASQA**: Introduced by (Ivan Stelmakh, Yi Luan, Bhuwan Dhingra and Ming-Wei Chang, et al., 2022), It is the first long-form question answering dataset that focuses on ambiguous factoid questions. Different from previous long-form answers datasets, each question is annotated with both long-form answers and extractive question-answer pairs, which should be answerable by the generated passage.

### 7.2 Data Preparation

The proposed method first involves mapping down continuous entities in a given video stream into discrete tokens through an encoder. Post that, during real time action estimation or targetting

editing, a decoder then provides relevant image information in the latent space which can be used for archival and editing purpose.

### 7.3 Algorithms & Techniques Description

#### 7.3.1 Targeted Variable Swapping

The intermediate variables in the U-Net of a diffusion model are proven to be informative about the content of the generated image. We focus on techniques such as localised editing wherein only the intended pixels are affected in operations such as image editing. Furthermore, we store the individual objects in the image in the form of latent image features obtained at each diffusion step of U-Net.

#### 7.3.2 Theory of Abstract Structures

According to the theory of Abstract Structures, any mathematical entity can be represented in the form of some basic structures such as Group, Rings, Modules and Fields. We model the latent feature representation tokens extracted above into these structures and also preserve the relationship between those tokens in these structures in a clear and concise manner. We prove that the proposed framework is capable of correctly storing these tokens in the given form. We intend to start with a bottom up approach wherein we formulate our construction in the simplest form possible, adding complexity as we go along.

#### 7.3.3 Church Turing Thesis

In the theory of computability, Church-Turing Thesis is a thesis around the class of computable functions that formally proves the existence of an effective method for calculating a function on the natural numbers if and only if it is computable by a Turing Machine. The significance of this result lies in the fact that we are going to use it as a basis to show that our proposed abstract structure construction can be effectively used in practical scenarios for archival and editing purposes on media content.

### 7.4 Implementation

The proposed implementation can be broadly separated into two major steps:

1. Tokenization – In this step, we break down a given input stream of media content into a stream of discrete tokens in a latent feature representation space. These tokens will capture all the information about the individual components of the media content as well as also the relationship between related components.
2. Archival into an Abstract Form- In this step, we will then store the discrete tokens and their underlying relationship in a binary text representation that can be directly worked upon for multiple downstream activities such as prompt answering, image editing, etc.

### 7.5 Evaluation

The proposed framework is evaluated mainly using analytical approaches. Since the construction is mathematical in nature, we use results from abstract algebra to first construct as well as prove the correctness of our proposed structure.

Post that, we then show using results from theory of computation that the proposed structure can be correctly implemented in the form of a turing recognizable language.

## 8. Required Resources

### 8.1 Hardware Requirements

The hardware requirements that must be met for this research work is as follows:

- A workstation with at least a dual core processor and 4 gb ram.
- A working internet connection with at least 50 mbps speed.
- A GPU for training and evaluating various models.

### 8.2 Software Requirements

The software requirements that must be met for this research work is as follows:

- Internet Browser such as Google Chrome.
- VS Code
- Python 3.7+
- C++ Compiler such as GCC
- Pytorch library.

### 8.3 Dataset Requirements

The following dataset requirements must be met for this research work:

- UCF11 dataset for action recognition tasks on video content from varying action categories.
- ASQA dataset for long-form question answering on ambiguous factoid questions.

# 9. Research Plan

## 9.1 Gantt Chart

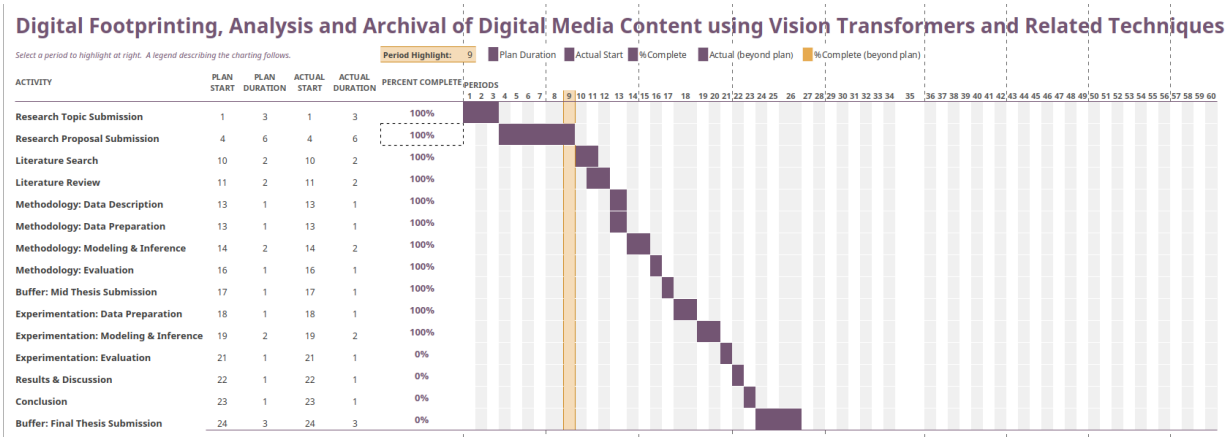
Digital Footprinting, Analysis and Archival of Digital Media Content using Vision Transformers and Related Techniques
Select a period to highlight at right. A legend describing the charting follows.
Period Highlight: 9
Plan Duration
Actual Start
%Complete
Actual (beyond plan)
%Complete (beyond plan)
ACTIVITY
PLAN START
PLAN DURATION
ACTUAL START
ACTUAL DURATION
PERCENT COMPLETE
PERIODS
Research Topic Submission 1 3 1 3 100%
Research Proposal Submission 4 6 4 6 100%
Literature Search 10 2 10 2 100%
Literature Review 11 2 11 2 100%
Methodology: Data Description 13 1 13 1 100%
Methodology: Data Preparation 13 1 13 1 100%
Methodology: Modeling & Inference 14 2 14 2 100%
Methodology: Evaluation 16 1 16 1 100%
Buffer: Mid Thesis Submission 17 1 17 1 100%
Experimentation: Data Preparation 18 1 18 1 100%
Experimentation: Modeling & Inference 19 2 19 2 100%
Experimentation: Evaluation 21 1 21 1 0%
Results & Discussion 22 1 22 1 0%
Conclusion 23 1 23 1 0%
Buffer: Final Thesis Submission 24 3 24 3 0%

**Figure 9.1.1**

**Note:** 1 Period = I Calendar Week

### 9.2 Risk Mitigation and Contingency Plan

Some of the potential risks that may show up later and the possible contingency plan is outlined below:

**Table 9.2.1**

| Risk | Contingency |
|---|---|
| Unplanned break due to health or unforeseen personal commitments | Proper buffer in project management<br><br>Inform the university in case of extension needed. |
| Unexpected hardware overrun for conducting the required experimentations. | Do a proper system requirement analysis upfront inorder to prevent any surprises later on. |

## References


Ali Hatamizadeh, Jiaming Song, Guilin Liu, Jan Kautz, Arash Vahdat (2024), DiffiT: Diffusion Vision Transformers for Image Generation. Available at: https://arxiv.org/abs/2312.02139

An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, Sifei Liu (2024), SpatialRGPT: Grounded Spatial Reasoning in Vision Language Models. Available at: https://arxiv.org/abs/2406.01584

Anran Liu, Cheng Lin, Yuan Liu, Xiaoxiao Long, Zhiyang Dou, Hao-Xiang Guo, Ping Luo, Wenping Wan), SpatialRGPT: Grounded Spatial Reasoning in Vision Language Models. Available at: https://arxiv.org/abs/2406.01584

Anran Liu, Cheng Lin, Yuan Liu, Xiaoxiao Long, Zhiyang Dou, Hao-Xiang Guo, Ping Luo, Wenping Wang (2024), Part123: Part-aware 3D Reconstruction from a Single-view Image. Available at: https://arxiv.org/abs/2405.16888

Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, Guiguang Ding (2024), YOLOv10: Real-Time End-to-End Object Detection. Available at: https://arxiv.org/abs/2405.14458

Bank, Dor; Koenigstein, Noam; Giryes, Raja (2023). "Autoencoders". In Rokach, Lior; Maimon, Oded; Shmueli, Erez (eds.). Machine learning for data science handbook. pp. 353–374.

Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, Lu Yuan (2023), Florence-2: Advancing a Unified Representation for a Variety of Vision Tasks. Available at: https://arxiv.org/abs/2311.06242g (2024), Part123: Part-aware 3D Reconstruction from a Single-view Image. Available at: https://arxiv.org/abs/2405.16888



Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, Guiguang Ding (2024), YOLOv10: Real-Time End-to-End Object Detection. Available at: https://arxiv.org/abs/2405.14458

Bank, Dor; Koenigstein, Noam; Giryes, Raja (2023). "Autoencoders". In Rokach, Lior; Maimon, Oded; Shmueli, Erez (eds.). Machine learning for data science handbook. pp. 353–374.



Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, Lu Yuan (2023), Florence-2: Advancing a Unified Representation for a Variety of Vision Tasks. Available at: https://arxiv.org/abs/2311.06242

Chang, Ziyi; Koulieris, George Alex; Shum, Hubert P. H. (2023). "On the Design Fundamentals of Diffusion Models: A Survey". Available at: https://arxiv.org/abs/2306.04542

Chong Mou, Mingdeng Cao, Xintao Wang, Zhaoyang Zhang, Ying Shan, Jian Zhang (2024), ReVideo: Remake a Video with Motion and Content Control. Available at: https://arxiv.org/abs/2405.13865

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Jonathan Larson (2024), From Local to Global: A Graph RAG Approach to Query-Focused Summarization. Available at: https://arxiv.org/abs/2404.16130

Fischer, Asja; Igel, Christian (2012), "An Introduction to Restricted Boltzmann Machines", Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, Lecture Notes in Computer Science, vol. 7441, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 14–36, doi:10.1007/978-3-642-33275-3_2, ISBN 978-3-642-33274-6

Gangwoo Kim, Sungdong Kim, Byeongguk Jeon, Joonsuk Park, Jaewoo Kang (2023), Tree of Clarifications: Answering Ambiguous Questions with Retrieval-Augmented Large Language Models. Available at: https://arxiv.org/abs/2310.14696

Gaurav Raut, Advait Patole (2023), End-to-End 3D Object Detection using LiDAR Point Cloud. Available at: https://arxiv.org/abs/2312.15377

Guilherme Potje, Felipe Cadar, Andre Araujo, Renato Martins, Erickson R. Nascimento (2024), XFeat: Accelerated Features for Lightweight Image Matching. Available at: https://arxiv.org/abs/2404.19174

Hinton G (2009). "Deep belief networks". Scholarpedia. 4 (5): 5947. Bibcode:2009SchpJ...4.5947H. doi:10.4249/scholarpedia.5947

Ivan Stelmakh, Yi Luan, Bhuwan Dhingra and Ming-Wei Chang, ASQA: Factoid Questions meet Long-Form Answers, Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pp. 8273-8288. Available at: https://arxiv.org/abs/2204.06092

Jingen Liu, Jiebo Luo and Mubarak Shah, Recognizing Realistic Actions from Videos "in the Wild", IEEE International Conference on Computer Vision and Pattern Recognition(CVPR), 2009. Available At: https://www.crcv.ucf.edu/papers/cvpr2009_liu1.pdf

Jing Gu, Nanxuan Zhao, Wei Xiong, Qing Liu, Zhifei Zhang, He Zhang, Jianming Zhang, HyunJoon Jung, Yilin Wang, Xin Eric Wang (2024), SwapAnything: Enabling Arbitrary Object Swapping in Personalized Visual Editing. Available at: https://arxiv.org/abs/2404.05717

Karen Simonyan, Andrew Zisserman (2014), Two-Stream Convolutional Networks for Action Recognition in Videos. Available at: https://arxiv.org/abs/1406.2199

Kingma, Diederik P.; Welling, Max (2022). "Auto-Encoding Variational Bayes". arXiv:1312.6114

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, Hengshuang Zhao (2024), Depth Anything V2. Available at: https://arxiv.org/abs/2406.09414

Lijun Zhou, Tao Tang, Pengkun Hao, Zihang He, Kalok Ho, Shuo Gu, Wenbo Hou, Zhihui Hao, Haiyang Sun, Kun Zhan, Peng Jia, Xianpeng Lang, Xiaodan Liang (2024), UA-Track: Uncertainty-Aware End-to-End 3D Multi-Object Tracking. Available at: https://arxiv.org/abs/2406.02147

Luyang Zhu, Yingwei Li, Nan Liu, Hao Peng, Dawei Yang, Ira Kemelmacher-Shlizerman (2024), M&M VTO: Multi-Garment Virtual Try-On and Editing. Available at: https://arxiv.org/abs/2406.04542

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, Christoph Feichtenhofer (2024), SAM 2: Segment Anything in Images and Videos. Available at: https://arxiv.org/abs/2408.00714

Ruiqi Gao, Aleksander Holynski, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul Srinivasan, Jonathan T. Barron, Ben Poole (2024), CAT3D: Create Anything in 3D with Multi-View Diffusion Models. Available at: https://arxiv.org/abs/2405.10314

Sai Kumar Dwivedi, Yu Sun, Priyanka Patel, Yao Feng, Michael J. Black (2024), TokenHMR: Advancing Human Mesh Recovery with a Tokenized Pose Representation. Available at: https://arxiv.org/abs/2404.16752

Sherrington, David; Kirkpatrick, Scott (1975), "Solvable Model of a Spin-Glass", Physical Review Letters, 35 (35): 1792–1796, doi:10.1103/PhysRevLett.35.1792

Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N; Kaiser, Łukasz; Polosukhin, Illia (2017). "Attention is All you Need" (PDF). Advances in Neural Information Processing Systems. 30. Curran Associates, Inc.

Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, Lifang He, Lichao Sun (2024), Sora: A Review on Background, Technology, Limitations, and Opportunities of Large Vision Models. Available at: https://arxiv.org/abs/2402.17177

Yuzhang Tian, Jianbo Zhao, Haoyu Dong, Junyu Xiong, Shiyu Xia, Mengyu Zhou, Yun Lin, José Cambronero, Yeye He, Shi Han, Dongmei Zhang (2024), SpreadsheetLLM: Encoding Spreadsheets for Large Language Models. Available at: https://arxiv.org/abs/2407.09025v1

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, Max Tegmark (2024), KAN: Kolmogorov-Arnold Networks. Available at: https://arxiv.org/abs/2404.19756